UNIVERSIDADE FUMEC
FACULDADE DE CIÊNCIAS EMPRESARIAIS
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JOÃO PAULO CARNEIRO ARAMUNI

DESENVOLVIMENTO ÁGIL DE APLICAÇÕES WEB

BELO HORIZONTE
2013

UNIVERSIDADE FUMEC

FACULDADE DE CIÊNCIAS EMPRESARIAIS

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JOÃO PAULO CARNEIRO ARAMUNI

DESENVOLVIMENTO ÁGIL DE APLICAÇÕES WEB

Trabalho de Conclusão de Curso apresentado à Universidade Fumec, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador Temático: Prof. Dr. Flávio Velloso Laper.
Orientador Metodológico: Prof. Msc. Wiliam Lopes Camelo.

BELO HORIZONTE

2013

UNIVERSIDADE FUMEC

FACULDADE DE CIÊNCIAS EMPRESARIAIS

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Trabalho de Conclusão de Curso defendido e aprovado/reprovado, em: ____/_____/2013, pela banca examinadora constituída por:

_____

Prof. Msc. William Lopes Camelo – Orientador Metodológico

_____

Prof. Dr. Flávio Velloso Laper – Orientador Temático

_____

BELO HORIZONTE

2013

"*Persistence is the shortest path to success*."

Charles Chaplin.

RESUMO

Esta pesquisa apresenta um estudo sobre o desenvolvimento de software para web, naturalmente ágil, com foco na produtividade, qualidade e segurança da aplicação desenvolvida. A difusão de informação ocasionada pela evolução da tecnologia e a crescente disseminação da internet como meio de comunicação tem levado os profissionais de tecnologia da informação (TI) a buscar novas abordagens de desenvolvimento de software. A demanda gradativa por aplicações robustas e escaláveis em nuvem tem forçado a comunidade científica a planejar novos modelos de desenvolvimento especificamente voltados para a web, dando lugar a uma nova área de conhecimento denominada “Engenharia Web”. A construção de software para web apresenta características exclusivas que não são encontradas na engenharia de software tradicional. Dessa forma, faz-se necessário entender as diferenças e os meios eficazes de se produzir software para a nuvem. Serão apresentados aqui, modelos e ferramentas de desenvolvimento ágil (frameworks), direcionados exclusivamente para a web, selecionados com base nos casos de sucesso e aceitação dentro do mercado de TI. O desenvolvimento web apresenta uma importante contribuição para a difusão do conhecimento, a evolução tecnológica, o crescimento econômico e o desenvolvimento cultural da sociedade.

Palavras-Chave: Engenharia Web, Engenharia de Software, Desenvolvimento, Frameworks, Metodologias Ágeis, Tecnologia.

ABSTRACT

This is a study of the web software development naturally agile, focused on productivity, quality and safety of the application developed. The evolution in technology and the increasing dissemination of internet as the main social network tool has led many information technology (IT) professionals to seek new approaches of software development. The gradual demand for robust and scalable applications in the cloud, has forced the scientific community to build new development models designed specifically for the web, introducing a new area of knowledge termed "Web Engineering". The web software development presents unique features that are not found in traditional software engineering. Thus, it is necessary to understand the differences and effective means of producing software for cloud. In this study we will show/ discuss fast / agile development models and tools focused on the web, based on succesfull and acceptable cases of IT field. The web development has an important contribution to the dissemination of knowledge, technological evolution, economic growth and society cultural development.

Key words: Web Engineering, Software Engineering, Development, Frameworks, Agile Methodologies, Technology.

.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API - Application Programming Interface
COC - Convention Over Configuration
CPU - Central Processing Unit
CRUD - Create, Read, Update e Delete
DRY - Don’t Repeat Yourself
ERP - Enterprise Resource Planning
FAQ - Frequently Asked Questions
GPU - Graphics Processing Unit
HTML - Hypertext Markup Language
HTTP - Hypertext Transfer Protocol
HTTPS - HyperText Transfer Protocol Secure
IDE - Integrated Development Environment
IP - Internet Protocol
ISO - International Organization for Standardization
JDK - Java Development Kit
JPA - Java Persistence API
MVC - Model-View-Controller
NAT - Network Address Translation
ORM - Object-Relational Mapping
SMTP - Simple Mail Transfer Protocol
SOAP - Simple Object Access Protocol
SQL - Structured Query Language
TI - Tecnologia da Informação
UI - User Interface
URL - Uniform Resource Locator
VM - Virtual Machine
VPC - Virtual Private Cloud
VPN - Virtual Private Network
XML - Extensible Markup Language

SUMÁRIO

1. INTRODUÇÃO

1.1. Contextualização do problema

O número de sistemas e aplicativos para a web tem crescido muito nos últimos anos, causando um grande impacto na história da computação. Com a sua importância aumentando, também se faz necessário o uso de uma abordagem disciplinada para a construção destes sistemas. Abordagens utilizadas na Engenharia de Software tradicional passaram a ser adotadas na construção de sistemas para a web.

Os aplicativos para a web são diferentes de outras categorias de software e têm características exclusivas: são dirigidos a conteúdo, estão em constante evolução, têm curto prazo de desenvolvimento, dentre outras.

Falta de planejamento, projetos mal feitos e falta de gerenciamento acabam tendo consequências sérias. Segundo Murugesan (2000), 84% dos sistemas entregues não atendem às necessidades do cliente; 79% dos projetos são entregues com atrasos e 63% têm custo maior que o orçamento previsto. Mais de 50% dos sistemas prontos são de baixa qualidade e faltam funcionalidades necessárias. (GINIGE, 2001).

Como resultado, desenvolvedores e usuários, começaram a se preocupar com a maneira como sistemas web complexos estão sendo criados, bem como com seus níveis de desempenho, qualidade e integridade.

1.2. Pergunta que direciona a pesquisa

Como desenvolver aplicações web de forma ágil, com foco na produtividade, sem perder em qualidade e segurança?

1.3. Objetivos

1.3.1. Objetivo Geral

Identificar as características que possibilitam um ambiente de desenvolvimento ágil de aplicações web seguras e de qualidade.

1.3.2. Objetivos Específicos

Investigar o ganho de produtividade oferecido por frameworks criados para atender as necessidades do desenvolvedor web, em cada linguagem, apontando características positivas e negativas de cada um.

Reconhecer modelos e padrões de desenvolvimento ágil para web.

Identificar os fatores que motivam organizações a criarem e/ou migrarem sistemas para web.

Entender o crescimento acelerado do Cloud Computing (Computação nas nuvens) alinhado ao desenvolvimento web e ao mercado de TI.

1.4. Justificativa

Segundo o IBGE (Instituto Brasileiro de Geografia e Estatística), a quantidade de internautas no Brasil aumentou 143,8% de 2005 para 2011. Um total de 47% da população conectada através de computadores, laptops, tablets e smartphones.

Pesquisa realizada pela IDC (International Data Corporation) encomendada pela Microsoft, afirma que a computação em nuvem será responsável pela criação de 14 milhões de novos empregos ao redor do mundo até 2015. As estimativas também mostram que, no Brasil, a computação em nuvem deve gerar o dobro de vagas de empregos ano a ano. Devido a essa explosão do Cloud Computing, a demanda por aplicações web pelo mercado é crescente, inclusive com a migração de sistemas legados locais para a nuvem.

A flexibilidade oferecida pela web, a demanda por serviços de internet móvel, bem como os altos custos para manter um sistema computacional capaz de hospedar as aplicações, são alguns dos fatores que motivam pequenas e médias empresas a investirem em aplicações na nuvem e terceirizarem sua hospedagem.

Uma vez reconhecida a necessidade de se desenvolver sistemas complexos e programáveis para a web, cabe à equipe de desenvolvimento definir metodologias eficazes que permitam a criação de software seguro e de qualidade para web. Sendo assim, é possível inserir um contexto de boas práticas e um ambiente de desenvolvimento ágil especificamente direcionado a web.

Na prática, o desenvolvimento ágil para web se traduz em utilização de frameworks como ferramentas de ganho de produtividade. Dessa forma, é crucial entender os prós e os contras de cada tipo de framework, entre os mais utilizados, para cada tipo de linguagem na web.

Essa pesquisa traz uma abordagem geral sobre os fatores que influenciam o desenvolvimento ágil de aplicações web. Dessa forma, este estudo não apresenta perfil de tutorial, e sim, perfil crítico para identificação de características que levam ao ganho de produtividade através de frameworks ágeis.

Este estudo referencia modelos de desenvolvimento ágil, mas não discute detalhes de sua implementação. Essa pesquisa não tem foco em processos, e sim em produtos. Apenas o conteúdo de importância para o desenvolvimento ágil na web será citado.

1.5. Hipótese

O desenvolvimento de aplicativos para a web é uma área relativamente nova e há poucos dados históricos que podem ser utilizados para fazer estimativas. Até agora, nenhum tipo de métrica foi publicado e ainda há pouca discussão de como devem ser estas métricas. Com isso, estimativas são baseadas apenas em experiências com projetos similares. Mas quase todo aplicativo para a web quer inovar em alguma coisa, oferecendo algo novo e diferente. Isto acaba fazendo com que estimativas baseadas em experiência com outros projetos, apesar de úteis, estejam sujeitas a uma alta margem de erro.

A rápida multiplicação da oferta de aplicações similares tornou a web bastante competitiva, e aspectos como confiabilidade, usabilidade e segurança passaram a serem considerados importantes diferenciais de qualidade. (OFFUTT, 2002).

Os desenvolvedores também tiveram que se ajustar à velocidade da web, realizando projetos em espaços de tempo menores, de apenas alguns meses, e adaptando suas aplicações com frequência para acompanhar as mudanças tecnológicas constantes, e atender às crescentes necessidades do mercado (GLASS, 2001).

O grande desafio então passa a ser adaptar as técnicas tradicionais existentes de Engenharia de Software, para uso na web criando assim o que é chamado de “Engenharia para a web”. A solução proposta por este trabalho refere-se então, a um estudo de tais técnicas e da maneira com que elas vêm sendo adotadas pelos desenvolvedores de sistemas para a web. Na teoria, a maioria das atividades de gerenciamento de projeto utilizadas na Engenharia de Software pode ser utilizada também na Engenharia para a web. Mas na prática, a abordagem é consideravelmente diferente.

Dentre as técnicas apresentadas, destaca-se a prática de modelos ágeis e a utilização de frameworks de produtividade como resposta ao problema de desenvolvimento de aplicações web seguras e de qualidade em curtos prazos.

2. REFERENCIAL TEÓRICO

2.1. Engenharia web

A web foi inicialmente concebida com o intuito de compartilhar informações científicas entre alguns poucos cientistas. O conteúdo era estático e apenas textual, não havia imagens, sons, animações ou conteúdo gerado dinamicamente para cada usuário, a interação era limitada, a navegabilidade era fácil, alto desempenho era desejável, mas não essencial, e os sites eram desenvolvidos por apenas uma pessoa ou um pequeno grupo.

A partir do início da década de 90, pode-se dizer que a internet tem permitido que a cada dia, mais e mais pessoas entrem no mundo da computação. Hoje é possível comprar ações e fundos mútuos, baixar músicas, ver filmes, vender artigos pessoais, reservar passagens aéreas, conhecer pessoas, fazer transações bancárias, assistir palestras e inúmeros outros usos que não poderiam ser imaginados dez anos atrás. Murugesan (2000) destaca que a internet levou apenas quatro anos para estar em 30% dos lares americanos. É um tempo bem curto quando comparado a outros produtos: o telefone levou 40 anos, o rádio levou 35 anos, o videocassete demorou 20 anos, a televisão 26 anos e o próprio computador levou 19 anos.

Com a popularização da tecnologia, a internet passou a ser utilizada não só como meio de disseminação de informações, mas também como plataforma de comunicação e desenvolvimento, e como canal para a realização de negócios. Essa mudança de contexto possibilitou o surgimento de duas novas modalidades de desenvolvimento de software: o Desenvolvimento de Aplicações web e o Desenvolvimento Global de Software. (DANTAS, 2003).

Com o tempo, escopo e complexidade foram aumentando, pequenos serviços foram cedendo espaço para grandes aplicativos, e com isso também aumentou a complexidade dos projetos e as dificuldades de desenvolvimento, manutenção, gerenciamento e controle de qualidade. Tudo feito com pouca disciplina, sem preocupação com técnicas e métodos padronizados, gerando aplicativos com grande probabilidade de problemas como baixo desempenho e falhas. Na web problemas como esses são ainda mais graves que no software tradicional, pois

como afirma Nielsen, “a distância entre um site e seus concorrentes é sempre de apenas poucos cliques”.

Com a difusão da internet móvel e o aparecimento do Wi-Fi em praticamente todos os centros urbanos, abre-se um leque de possibilidades para atender um novo tipo de mercado dinâmico na web. Seja para E-commerce, internet Banking, ou Redes Sociais, o uso de tecnologias voltadas para a web é cada vez mais comum e faz parte do cotidiano da sociedade.

Em se tratando de organizações a web oferece flexibilidade e independência de plataformas, uma vez que o acesso a programas, serviços e arquivos é remoto, através da internet. Como grande parte do processamento é feito do lado servidor, a máquina cliente não precisa de configuração robusta, sistema operacional padronizado ou programas específicos instalados. Basta um navegador para então poder utilizar desde aplicações pequenas até ERP’s parrudos. Em contrapartida a web demanda preocupações no desenvolvimento, como segurança de dados e desempenho.

Uma aplicação web deve desempenhar diversas tarefas distintas como, possuir uma interface com boa usabilidade para o usuário, apresentar os dados de forma clara e objetiva, controlar o fluxo de páginas da aplicação, interagir com o usuário, proteger os dados que trafegam durante o clico de vida da sessão, implementar a lógica de negócios e prover acesso rápido aos dados armazenados. As tecnologias de acesso a dados flexíveis e eficazes são a alma dos sites e aplicações web dinâmicos.

Segundo Pressman (2006), sistemas e aplicativos da web são caracterizados por disponibilizar grande quantidade de conteúdo e funcionalidade para grande população de usuários.

O Desenvolvimento de Aplicações web mostrou-se uma excelente oportunidade de negócio para as empresas de software, que poderiam oferecer serviços a usuários finais de todo o mundo. As aplicações atuais servem a uma variedade de propósitos, desde a troca de mensagens em listas de discussão e a participação em jogos on-line, à realização de negócios de comércio eletrônico e transações bancárias (GINIGE, 2001).

Mas, para que isso fosse possível, foi necessária uma mudança nas características e na arquitetura das aplicações web. Enquanto as primeiras aplicações se resumiam a um conjunto de poucas páginas com conteúdo estático e,

principalmente textual, as aplicações atuais são maiores e mais complexas, com páginas que apresentam conteúdo multimídia e dinâmico, e permitem acessos a bancos de dados e comunicação com sistemas legados. (MURUGESAN, 2001).

A Engenharia para a web é, portanto, o processo utilizado para criar aplicativos web de alta qualidade. A Engenharia para web não é igual à Engenharia de Software tradicional, mas ambos compartilham muitos conceitos e princípios fundamentais, com ênfase nas mesmas técnicas de gerenciamento e atividades. Há pequenas diferenças na maneira como essas atividades são conduzidas, mas a filosofia que dita uma abordagem disciplinada para o desenvolvimento de um sistema de computador é a mesma.

De acordo com Ginige (2001), a construção de um sistema para a web necessita do conhecimento de pessoas de diferentes áreas. Como resultado, a Engenharia web é multidisciplinar, e dela participam áreas como: análise de sistemas e projetos; engenharia de software; engenharia de hipermídia e hipertexto; engenharia de requisitos; interação homem-máquina; desenvolvimento de interface de usuário; engenharia de informação; indexação e recuperação de informações; teste; modelagem e simulação; gerenciamento de projetos; e projeto gráfico e apresentação.

A engenharia web aplica uma abordagem genérica combinada com estratégias, táticas e métodos especializados. O processo de engenharia web começa com uma formulação do problema a ser resolvido pela web. O projeto é planejado e os requisitos são analisados. Os projetos arquitetural, navegacional e de interface são conduzidos. O sistema é implementado usando linguagens e ferramentas especializadas, associadas com a web, e iniciam-se os testes. Como os aplicativos para a web evoluem continuamente, são necessários mecanismos para controle de configuração, garantia de qualidade e contínuo suporte a esta evolução.

A engenharia de web diz respeito ao estabelecimento e uso de princípios científicos sólidos, de engenharia e de gestão, e abordagens disciplinadas e sistemáticas para que o desenvolvimento, a disponibilização e a manutenção de sistemas e aplicações de alta qualidade baseados na web sejam bem sucedidos.

Pressman (2006) conclui que as aplicações web envolvem uma mistura de publicação impressa e desenvolvimento de aplicações, de comercialização e computação, de comunicações internas e relações externas, e de arte e tecnologia.

2.2. Desenvolvimento Ágil

"A maioria de nossas suposições sobre negócios, tecnologia e organizações, tem pelo menos 50 anos de idade. Elas sobreviveram a seu tempo. Como resultado, nós estamos ensinando e praticando políticas que estão cada vez mais em desacordo com a realidade e, portanto, contraproducentes." (DRUCKER, 1998).

A tecnologia da informação e a inteligência de negócio vivem, na grande maioria do mercado, em constante conflito de interesses. Por um lado, os gastos com a TI são vistos como despesas e não como investimento. Por outro, a TI visualiza os administradores de negócios como completos alienados em tecnologia. A comunicação entre TI e negócio é um desafio atual com denominação própria, "Governança de TI".

Abaixo, Fig.1, apresenta uma sátira de como os usuários enxergam os programados e vice-versa. A comunicação escassa é um dos muitos problemas enfrentados pela Governança de TI. A sátira demonstra um problema comum em ambientes de negócio.

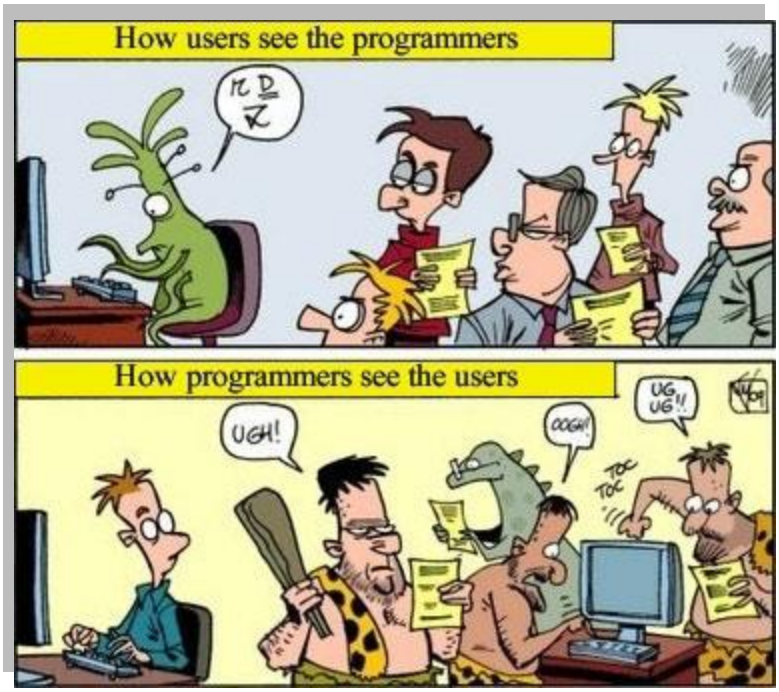

Figura 1: Como os usuários veem os programadores e vice-versa.

Fonte: 9GAG. Disponível em: http://9gag.com/gag/372032. Acesso em: 29/08/2013.

As metodologias tradicionais de processos e controle de negócio não acompanham a rápida evolução e dinamismo da tecnologia. Sendo assim, faz-se necessário o uso de novos modelos processuais mais modernos e flexíveis que consigam atender as necessidades do negócio, sem engessar o departamento de tecnologia da informação.

Existem diversos tipos de metodologias ágeis flexíveis, como: SCRUM, EUP (Enterprise Unified Process) e Lean Manufacturing da Toyota, que apoiam as decisões de negócio e são adaptáveis a qualquer tipo de projeto, porém este tópico irá focar-se em metodologias de processos especificamente voltadas para a construção de software.

Em se tratando de modelos de desenvolvimento de software, processos que tentam descrever minuciosamente cada passo da construção de um software são a base das metodologias tradicionais. A crescente demanda por sistemas e a alta velocidade com que seus requisitos evoluem têm evidenciado que desenvolver software exige flexibilidade, uma vez que muitas decisões precisam ser tomadas durante o projeto. Além disso, as dificuldades para a produção de sistemas vão muito além das questões técnicas. Fatores estratégicos, comerciais e humanos são responsáveis por algumas das variáveis que contribuem para tornar o desenvolvimento de sistemas de software uma atividade altamente complexa.
Modelos tradicionais de desenvolvimento de software propõem processos *prescritivos* que não consideram toda essa complexidade. (BASSI, 2008).

O objetivo por trás dessa rigidez é facilitar a replicação do processo em diversos ambientes, pois ele trata as pessoas como recursos facilmente substituíveis. Assim, o processo eliminaria, na teoria, o risco associado a erros humanos, pois as tarefas seguiriam as especificações determinadas no processo.

A inconsistência desta abordagem prescritiva é a tentativa de tornar uma atividade inerentemente feita com a criatividade humana em uma atividade mecanicamente replicável.

Se o desenvolvimento de software fosse uma operação determinística, as pessoas que participam do processo poderiam ser substituídas por máquinas ou programas de computador. Os frameworks que serão apresentados nesta pesquisa, não têm objetivo de substituir pessoas, mas auxiliar em tarefas que são frequentemente repetidas, poupando o desenvolvedor do retrabalho.

Métodos ágeis apoiam-se em poucas regras rigidamente definidas e em princípios que oferecem certa flexibilidade para que as práticas do dia-a-dia sejam adaptadas. O modelo ágil não descarta a estruturação e documentação do projeto de software, mas enfatiza preferencialmente a comunicação entre os stakeholders (todos os envolvidos com o projeto) em tempo real. Por causa disso, os métodos ágeis são frequentemente caracterizados como metodologias não-disciplinadas ou não-planejadas.

O desenvolvimento ágil de software sugere uma abordagem mais humanística com foco na entrega rápida e constante de software com valor de negócios. Porém, para conseguir isto, é preciso escolher um conjunto de práticas de desenvolvimento adequado às características de projeto e da equipe. (BASSI, 2008).

No contexto ágil, os processos caracterizam-se como empíricos e adaptativos. A principal influência para os processos ágeis é o modelo de melhoria continua de qualidade que Edwards Deming criou nos anos 50 baseado no modelo estatístico de controle de qualidade proposto por Walter Shewhart na década de 20.

A maioria dos métodos ágeis tenta minimizar os riscos do desenvolvimento de software através de curtos períodos de trabalho chamados de “iteração”. Dessa forma, a gestão e o controle dos processos de software se tornam mais simplificados e objetivos. Abaixo, Fig.2, Incremental x Iterativo. Diferença entre as duas abordagens.

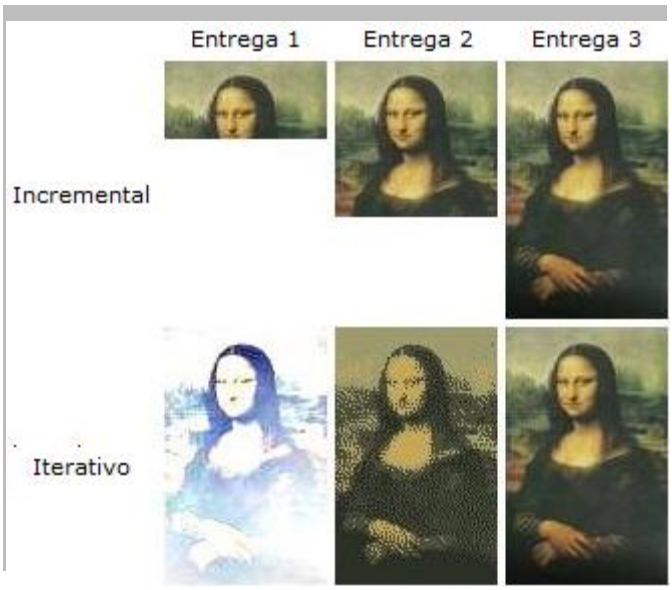

Figura 2: Incremental x Iterativo.

Fonte: Agile Way. Disponível em: http://www.agileway.com.br/2009/08/18/incremental-vs-iterativo/. Acesso em: 10/10/2013.

O objetivo do modelo incremental é evitar o retrabalho, porém o problema em usar esta técnica é que ela exige do cliente uma visão muito assertiva do que ele realmente deseja, mas nem sempre o cliente tem total informação e visão do que ele deseja no sistema. Nos modelos ágeis, a comunicação constante entre a equipe de desenvolvimento e o cliente, durante as iterações, evita a geração de mudanças após as entregas do sistema, uma vez que o cliente acompanha todo o processo de desenvolvimento e as mudanças acontecem de forma dinâmica e natural.

Princípios do manifesto ágil:

- Adaptação e resposta dinâmica a mudanças mais que seguir um plano.
- Colaboração com clientes mais do que negociação de contratos.
- Indivíduos e iterações mais do que processos e ferramentas.
- Software funcional mais que documentação extensa.

Apesar de haver importância nos itens à direita, o manifesto ágil prioriza os itens à esquerda. (Manifesto for Agile Software Development, 2001).

São características básicas do modelo ágil: Simplicidade; Motivação; Cooperação entre Stakeholders; Entregas Frequentes; Satisfação do Cliente.

Processos, contratos, documentação e planejamento têm valor para o desenvolvimento, mas são menos importantes do que saber lidar com pessoas, do que ter o cliente colaborando para encontrar a melhor solução, do que entregar o software com qualidade, do que se adaptar as mudanças (KALERMO, 2002).

Metodologias de desenvolvimento ágil são perfeitamente adaptáveis para web e aceitam a utilização de frameworks e de outras ferramentas de produtividade. Nesse ponto, a construção de software para web, se assemelha muito à construção de software para Desktop.

As metodologias, então chamadas de ágeis, propõem a obtenção de resultados práticos em um período curto de tempo, tirando o foco do processo e colocando no produto. Para isso, foi preciso que os métodos ágeis dispensassem ou modificassem as etapas do processo e a forma como os envolvidos com o desenvolvimento realizam suas atividades. Muitas dessas mudanças alteram características tidas como essências pelos métodos tradicionais, por isso as abordagens ágeis tornaram-se polêmicas e não inspiram confiança nos mais conservadores.

2.3. Padrão MVC

2.3.1. Padrão de Projeto

Padrão de Projeto é uma solução reutilizável, eficaz, para resolver problemas recorrentes de desenvolvimento de software. É um modelo capaz de oferecer um seguimento de construção e facilitar o entendimento do sistema.

Para Fowler, (1997), os padrões descrevem maneiras comuns de fazer as coisas e são coletados por pessoas que identificam temas repetitivos em projetos. Essas pessoas identificam cada tema e o descrevem de modo que outras pessoas possam ler o padrão e ver como aplica-los.

O Padrão MVC é um modelo de desenvolvimento considerado um padrão de projeto que separa a lógica de controle, o modelo de dados e a interface do usuário de uma aplicação em três componentes distintos. Dessa forma é possível desenvolver um componente com impacto mínimo sobre os demais.

O padrão MVC foi descrito pela primeira vez em 1979 por Trygve Reenskaug, que trabalhava no Smalltalk, na Xerox PARC. A implementação original é descrita em profundidade no artigo "Applications Programming in Smalltalk-80: How to use Model–View–Controller".

O MVC provê uma forma organizada de encapsular as tarefas da aplicação, de modo que a (re) implementação de uma delas possa ser feita sem afetar as demais. Em um ambiente com muitos profissionais trabalhando em uma mesma aplicação, esse tipo de separação lógica pode ser extremamente útil ao desenvolvimento. É possível que desenvolvedores trabalhem nas regras de negócio da aplicação enquanto, em paralelo, designers trabalham na exibição e interface com o usuário.

Para alcançar tal nível de organização e flexibilidade, o MVC divide a aplicação em três áreas de conteúdo distintas:

- Model (Modelo).
- View (Visualização).
- Controller (Controlador):

2.3.2. Modelo

O **Modelo** é a coleção de dados. Ex.: Cliente, Vendedor, Fornecedor, etc. Um modelo, ou Model, armazena os dados e encapsula a lógica de negócios da aplicação. Ele desconhece completamente a Visualização e o Controlador.
Funcionalidades do Modelo:

- Recuperar e armazenar dados no Banco de Dados.
- Formatar os dados para fácil acesso da Visualização.
- Implementar a camada de lógica de negócios que valida e consome os dados de entrada da Visualização.

Particularidades:

- O Modelo atua como um nível de abstração entre os dados de negócio e a interface que os exibe.
- É responsável por acessar dados, fornecer serviços de negócio e persistir objetos de negócio.
- O Modelo deve ser independente, não deve fazer referências nem a Visualização, nem ao Controlador.
- Em geral, o Modelo é implementado, para Java, com POJO's (Plain Old Java Objects, objetos Java que seguem um padrão de desenho e possuem em sua estrutura um construtor sem argumentos e métodos getters e setters para seus atributos.) ou EJB's (Enterprise JavaBeans, componentes da plataforma Java EE que rodam em um container de um servidor de aplicação.).

2.3.3. Visualização

A **Visualização** é responsável pela renderização e exibição dos dados provindos do Modelo. Quando o usuário executa uma ação em uma View, um evento será submetido ao Controlador, que decidirá a ação a ser tomada. A Visualização tem acesso livre ao Modelo e aos métodos de query fornecidos. Uma Visualização não deve alterar o estado de um Modelo e sim utiliza-lo para obtenção de dados.

A Visualização não tem controle sobre a lógica de navegação da aplicação, isso pode depender de outras lógicas relacionadas a ações do usuário.

A implementação de uma Visualização, ou View, pode ser realizada por JSP (JavaServer Pages), JSF (JavaServer Faces), Velocity, Tapestry, etc.

Para um desenvolvimento mais elegante, com boas práticas de programação, deve-se evitar o uso de scriplets diretamente no corpo dos JSP's e dar preferência ao uso de taglibs para, entre outras funções, instanciar objetos.

As páginas individuais que constituem a Visualização não têm qualquer conhecimento de suas relações com outras páginas. Elas simplesmente geram eventos para o Controlador. Existe uma independência entre páginas.

A Visualização e o Controlador não têm conhecimento da implementação das estruturas de dados da aplicação ou das regras de negócio, apenas utilizam a interface fornecida pelo Modelo.

2.3.4. Controlador

O **Controlador**, ou Controller, recebe e traduz entradas e requisições para o Modelo ou para a Visualização. É responsável pela chamada de métodos do modelo que alteram o estado do mesmo.

Qualquer interação do usuário com o sistema é submetida ao Controlador, que decide a ação a ser tomada. (Ex.: Chamada de método.).

Um Controlador é tipicamente implementado através de diversos Servlets e POJO's, responsáveis por diferentes aspectos de sua funcionalidade.

Abaixo, Fig. 3: Esquema do Padrão MVC:

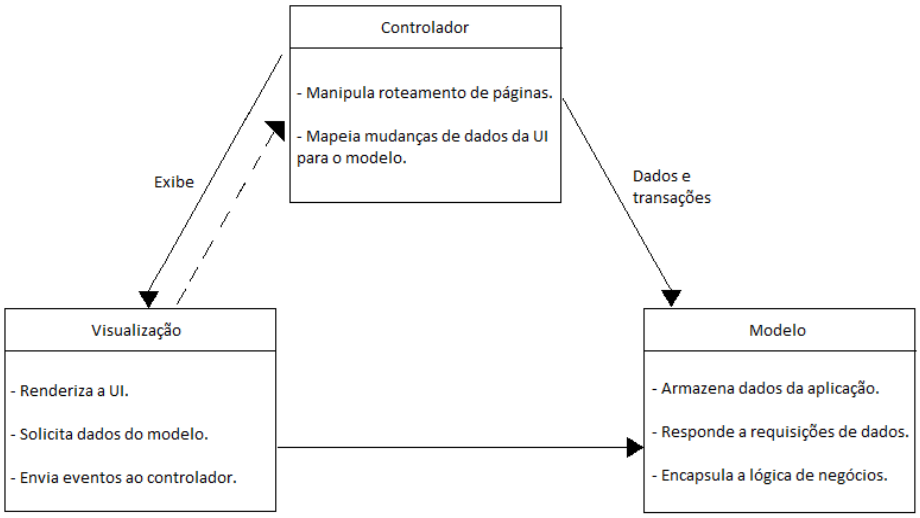

Figura 3: Padrão MVC.

Fonte: Material base da disciplina de Desenvolvimento web da Universidade Fumec.

2.3.5. Fluxo de Trabalho

O controle do fluxo realizado pelo Padrão MVC segue uma lógica simples:

- Interação do usuário com a interface. (Ex.: Inserção de dados e submissão de um formulário.).
- Manipulação do evento de entrada pelo Controlador, comumente feito através de um handler ou callback registrado.
- Acesso do Controlador ao Modelo para solicitação/alteração de dados.
- Obtenção de dados do Modelo pela Visualização. O controlador realiza as ações necessárias e solicita a Visualização a interface apropriada.
- Estado de espera da Interface por novas interações com o usuário.

Apesar de desenvolvido originalmente para computação pessoal, o MVC foi amplamente adotado como uma arquitetura para aplicações web. Muitos frameworks que aplicam o modelo foram criados nesse contexto, diferindo apenas na distribuição de cargas de trabalho entre cliente e servidor.

Os frameworks MVC mais modernos possuem uma abordagem “thin client”, onde todo processamento pesado é realizado no lado servidor. Nessa abordagem, o cliente envia uma requisição para o servidor, o controlador decide como agir e então um documento da visão, que solicita dados do modelo, é gerado dinamicamente para apresentar a saída de dados ao usuário.

2.4. Frameworks

2.4.1. Motor de Produtividade

Segundo Johnson (1997), um framework é um conjunto de classes que incorpora um projeto abstrato de soluções para uma família de problemas conhecidos.

Fazendo uma analogia com essa pesquisa, um framework poderia oferecer serviços de formatação de texto, verificação de normas da ABNT (Associação Brasileira de Normas Técnicas) e revisão de ortografia. Dessa forma, o pesquisador iria se focar apenas com a obra a ser escrita.

Uma das propostas da reutilização aceita na área da orientação a objetos é a de usar frameworks para domínios específicos, que poderiam ser instanciados para produzir novos produtos no mesmo domínio. (FERREIRA, 2006). Pode-se entender a reutilização de código como um processo de construir sistemas a partir de artefatos de outros sistemas existentes, ao invés de construí-los a partir do zero.

Um framework deve facilitar a construção de novos componentes, fornecer uma interface padrão para os mesmos trocarem dados, manipularem erros e chamarem operações. Os frameworks oferecem serviços já implementados (frozen spots) que normalmente realizam chamadas indiretas as funcionalidades e serviços que ainda serão implementados (host spots). Ao utilizar um framework, o trabalho do desenvolvedor consiste apenas em prover as partes que são específicas para sua aplicação (lógica de negócio).

Segundo Fayad (1999), frameworks apresentam os seguintes benefícios:

- Modularização: Encapsulamento dos detalhes voláteis de implementação através de interfaces estáveis.
- Reutilização: Definição de componentes genéricos que podem ser replicados para criar novos sistemas.
- Extensibilidade: Favorecida pelo uso de métodos hooks que permitem que as aplicações estendam interfaces estáveis.
- Inversão de controle – IoC : O código do desenvolvedor é chamado pelo código do framework. Dessa forma, o framework controla a estrutura e o fluxo de execução dos programas.

Para atingir alta produtividade no desenvolvimento web, está surgindo uma série de frameworks construídos a partir de conceitos como. “Convention Over Configuration” e “Kiss (Keep It Simple, Stupid!).”.

Isso significa que, se o desenvolvedor seguir as convenções ditadas pelo framework, o mesmo será capaz de inferir as ações a serem tomadas, sem a necessidade de configurações adicionais, como XML, Annotations ou qualquer outro tipo de meta-dado.

Além disso, o framework pode gerar vários templates pré-definidos de aplicações e boas práticas, como suporte a testes unitários, etc.

Ao poupar o programador de configurações complicadas, do gerenciamento de arquivos XML, e de boa parte do que se refere à infraestrutura, o mesmo pode se focar mais no core bussiness (núcleo de negócio) da aplicação, com efetivo ganho de qualidade e produtividade.

O processo de construção de um framework é muito mais complexo do que de uma aplicação tradicional, devido à flexibilidade inerente e a capacidade de variação de um framework. Construir um framework, não é uma tarefa fácil. O caminho até a reutilização prometida é árduo, mas os benefícios justificam os desafios. A Fig. 4 abaixo demonstra as situações que motivam ou não a criação de um framework. A, B e C, representam aplicações distintas com funcionalidades em comum.

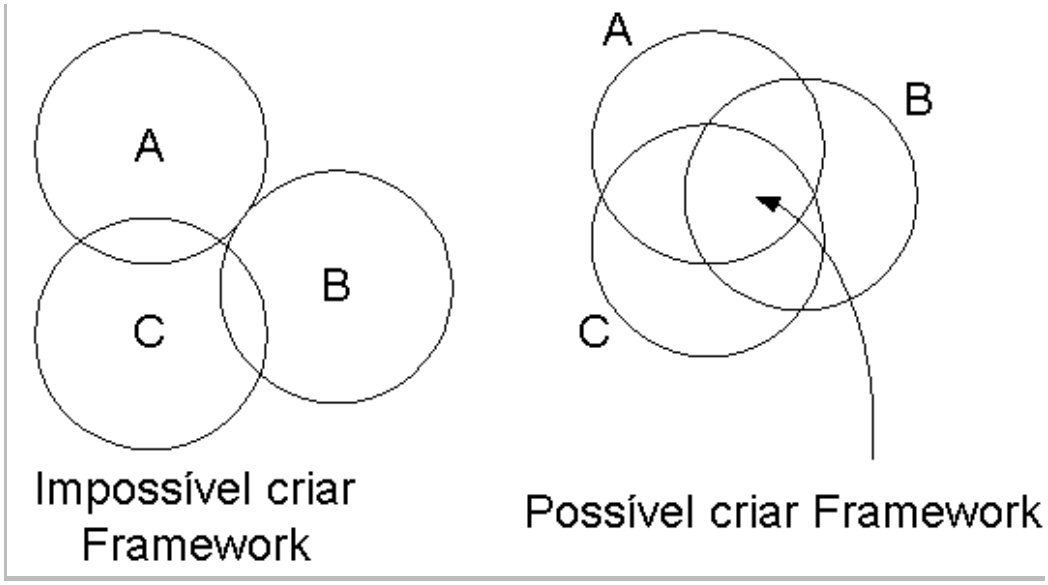

Figura 4: Interseção para criar frameworks.

Fonte: UFCG - (Universidade Federal de Campina Grande).

Disponível em: http://www.dsc.ufcg.edu.br/~jacques/cursos/. Acesso em: 15/08/2013.

Pensando em orientação a objetos, um framework utiliza um conjunto de classes e interfaces que mostra como decompor a família de problemas, e como objetos dessas classes colaboram para cumprir suas responsabilidades. O conjunto de classes deve ser flexível e extensível para permitir a construção de várias aplicações com pouco esforço, especificando apenas as particularidades, regras de negócio, de cada aplicação.

O framework é usado de acordo com o “Hollywood Principle” ("Don't call us, we'll call you"). É o framework que chama o código da aplicação (que trata das particularidades dessa aplicação), controla o fluxo de execução, define interações entre objetos e provê um comportamento padrão. Diferentemente de uma biblioteca de classes onde o cliente é quem chama as funções, não há controle de fluxo e nenhuma interação ou comportamento pré-definido. Abaixo, a Fig. 5 ilustra o Upside-down library:

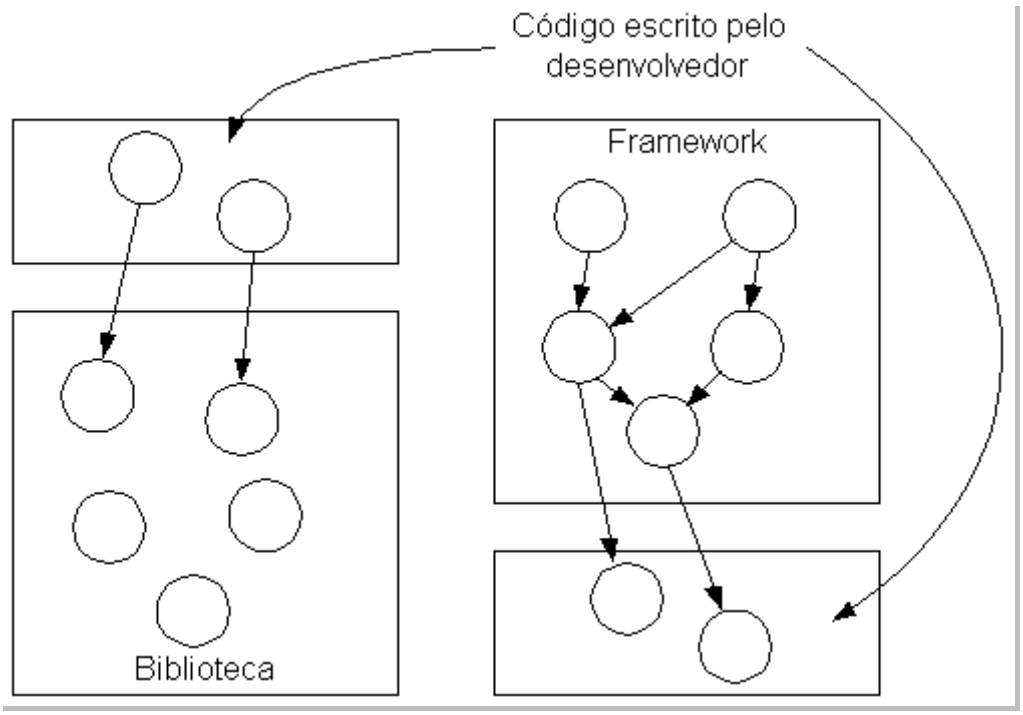

Figura 5: Fluxo entre framework e aplicação.

Fonte: UFCG - (Universidade Federal de Campina Grande).

Disponível em: http://www.dsc.ufcg.edu.br/~jacques/cursos/. Acesso em: 15/08/2013.

Um conceito muito próximo ao dos frameworks é o de padrões de projeto, design patterns, o que frequentemente confunde o desenvolvedor sobre o que

caracteriza cada conceito e quais são as suas particularidades, semelhanças e relações.

Um framework não pode ser considerado como um padrão de projeto. Aparentemente, os dois consistem de classes, interfaces e colaborações prontas, porém padrões de projeto são mais abstratos e genéricos do que frameworks. Como diferença básica, um framework inclui código e pode ser executado em nível de código, um padrão de projeto não.

Padrões de projeto são elementos arquiteturais menores e são menos especializados/direcionados que frameworks. Um framework típico contém vários padrões de projeto, mas o contrário nunca ocorre.

Padrões documentam frameworks e ajudam a garantir o uso correto de sua funcionalidade. Para Gamma, (1995) o objetivo do uso de padrões é capturar a experiência de projetos de software de forma que os projetistas possam usá-la efetivamente. Além disso, Johnson (1997) argumenta que a uniformidade reduz o custo da manutenção de software, pois, os desenvolvedores podem migrar mais facilmente entre aplicações sem necessitar de aprender tudo de novo.

Concluindo, um framework é uma tecnologia emergente e promissora para suportar reutilização de software em grande escala. Deve ser construído fazendo uso de padrões de projetos para melhorar sua documentação e sua arquitetura proporcionar um vocabulário simples e único entre desenvolvedores. Deve ser especializado, usável, reusável, bem documentado, extensível, seguro, eficiente e completo.

Em se tratando de web, a sociedade da informação não está mais em um ambiente de Hipertexto estático. Há um novo cenário de sites totalmente dinâmicos e interligados que proporcionam um habitat natural para o surgimento de frameworks ágeis.

O mercado de fabricação de sistemas para web percebeu as vantagens da utilização de frameworks. Hoje, são considerados pré-requisitos de experiência para alocação de profissionais em projetos.

2.4.2. Análise de Frameworks Populares

Neste tópico serão analisados e comparados seis frameworks criteriosamente selecionados com base em linguagens mais utilizadas no mercado, popularidade entre desenvolvedores, usabilidade e produtividade. Todos os frameworks apresentados são para desenvolvimento web e seguem o padrão MVC.

Essa abordagem é introdutória e teórica, não possui perfil didático (Explicação de comandos, funções...). Os frameworks escolhidos serão avaliados em suas particularidades e características.

2.4.2.1. Play Framework (JAVA/SCALA)

O Play é um framework open source para aplicações web, escrito em Java e Scala, que possibilita o desenvolvimento de aplicações web que seguem o padrão MVC.

Tem por objetivo otimizar a produtividade do desenvolvedor através do uso de configuração sobre convenção (CoC). Com recompilação feita durante a execução da aplicação, e caso ocorra algum erro, este é exibido no browser, com a indicação da linha do erro.

Apesar de o Play ter sido escrito em Java, ele suporta a linguagem Scala desde a versão 1.1 (Atual: 2.1.1). A empresa Typesafe (responsável pela linguagem Scala), anunciou a aquisição do Play Framework e este será mantido pela empresa. Uma das novidades relacionadas a este anúncio é a reescrita do núcleo do Play Framework, na versão 2.0, totalmente em Scala.

O Play foi muito inspirado no Ruby on Rails e Django. Um desenvolvedor familiarizado com qualquer um desses frameworks, irá se adaptar ao Play com facilidade.

O Play Framework utiliza do poder das aplicações Java, porém sem a burocracia necessária para o desenvolvimento de aplicações centradas no modelo Java Enterprise. Libertando das metodologias e ideologias relacionadas ao desenvolvimento de aplicações Java EE, o Play provê para os desenvolvedores uma maneira fácil e elegante de trabalhar, visando o aumento da produtividade.

Basta um editor de texto e será mais que o suficiente para o desenvolvimento de aplicações, chega a ser incrível pensar que é possível desenvolver aplicações

Java web sem a necessidade de um IDE (Eclipse, Netbeans,…), mas vale lembrar, que estas IDE's ainda possuem seus atrativos e auxiliam na produtividade do desenvolvedor.

Apesar que as aplicações desenvolvidas com o Play tenham sido projetadas para executar dentro do JBoss Netty web Server, estas podem ser empacotadas em arquivos WAR e distribuídas para outros servidores de aplicações Java EE (ex.: Apache Tomcat). Abaixo, Quadro 1: Diferenciais do Play Framework. Apresentação das principais características do Framework. Fonte: Play Framework. Disponível em: http://playframework.org/. Acesso em: 21/10/2013.

Quadro 1: Diferenciais do Play Framework.

Stateless: O Play é totalmente RESTful – não existe conexão por sessão Java EE. Isto torna o Play muito mais escalável que os demais frameworks.
Sem configuração: realizar o download, descompactar e desenvolver.
Fácil ida e volta: sem necessidade de deploy no servidor de aplicação, apenas edite o código e atualize o browser.
Teste unitário integrado: suportes nativos para JUnit e Selenium.
API elegante: raramente um desenvolvedor terá a necessidade de importar alguma lib. O Play já disponibiliza a maioria dos recursos necessários para o desenvolvimento de uma aplicação.
Métodos estáticos: todos os controles de entrada e métodos de negócio são declarados como estáticos. Isto é bem diferente do que se vê nos demais frameworks Java.
I/O Assíncrona: através do uso do servidor web JBoss Netty, o Play consegue disponibilizar e tratar uma enorme quantidade de requisições assíncronas.
Arquitetura Modular: assim como Rail e Django, o Play utiliza o conceito de módulos. O que possibilita um meio elegante e simples de expandir o core do Play.
Módulo CRUD: fácil construção de UI administrativas com pouco código.
Módulo Scala: disponibiliza um suporte completo para Scala (Linguagem Funcional. Exemplo de Aplicação web que tem seu back-end escrito em Scala: Twitter.).

Em se tratando de componentes, o Play utiliza algumas bibliotecas populares:

- JBoss Netty para o servidor web.
- Hibernate 2.0 para a camada de dados.
- Groovy para a os templates.
- Apache Ivy para gerenciamento de dependências.
- O compilador do Eclipse para atualização da aplicação sem necessidade de realizar um deploy da aplicação para testar as alterações (hot-reloading).

Abaixo, Quadro 2: Funcionalidades presentes no núcleo do Play. Apresentação das principais funções da ferramenta. Fonte: Play Framework. Disponível em: http://playframework.org/. Acesso em: 21/10/2013.

Quadro 2: Funcionalidades presentes no núcleo do Play.

Um framework RESTful limpo e leve.
CRUD: um módulo para simplificar a edição de modelos de objetos.
Secure: um módulo para habilitar um sistema de autenticação de usuários.
Um framework de validação baseado em anotações.
Um Job Scheduler (agendamento de tarefas).
Suporte para emails SMTP de maneira simples.
Suporte para JSON e XML.
Uma camada de persistência baseada em JPA.
Uma base de dados embutida para rápido deploy e testes da aplicação
Um framework completo para realização de testes.
Funcionalidade para upload de arquivos.
Suporte para múltiplos ambientes de desenvolvimento.
Poderosa engine de templates baseadas em Groovy com hierarquia e tags.
Arquitetura modular que possibilita criar novas funcionalidades para o núcleo.
Suporte para OpenID e clientes web Service.

Quem está acostumado com os frameworks tradicionais do mercado, deve imaginar que é necessário criar um projeto web em um IDE, importar alguns arquivos jars, configurar certos arquivos e escrever a aplicação, entretanto com o Play Framework, esse processo se torna incrivelmente simples. Para criar uma nova aplicação, usa-se um utilitário, chamado **Play Console**:

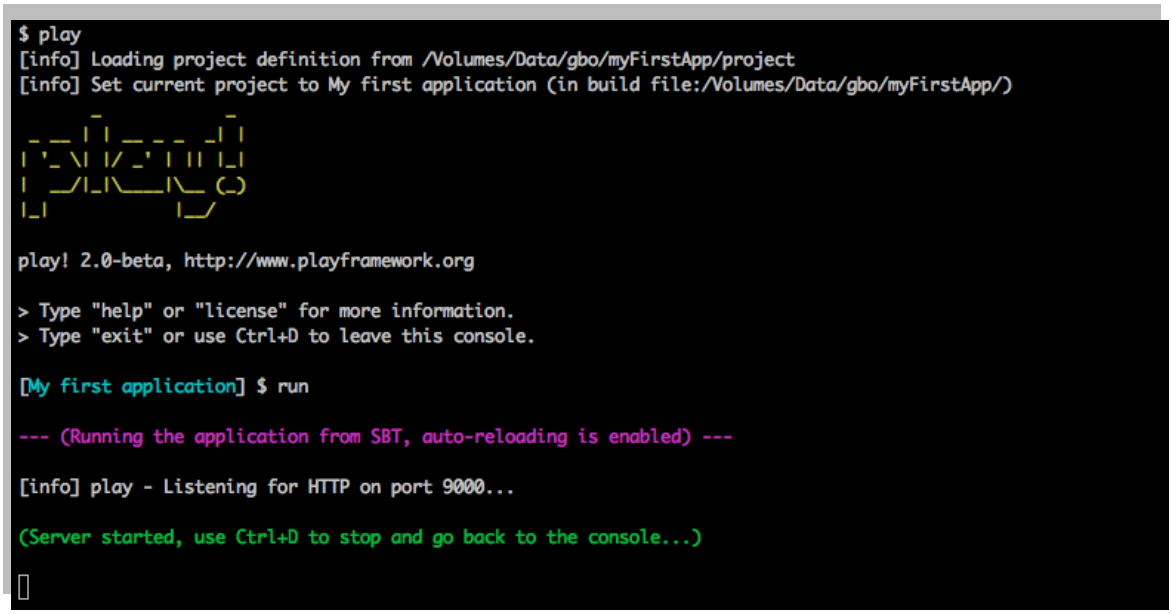

Figura 6: Play Console.

Fonte: Play! . Disponível em: http://www.playframework.com/documentation/2.0/PlayConsole. Acesso em: 15/08/2013.

Outra grande funcionalidade do Play Framework é a manipulação de erros diretamente no Browser. Com o erro na tela, podemos facilmente corrigir e recarregar a página sem precisar recompilar nada. Abaixo, exemplo de erro:

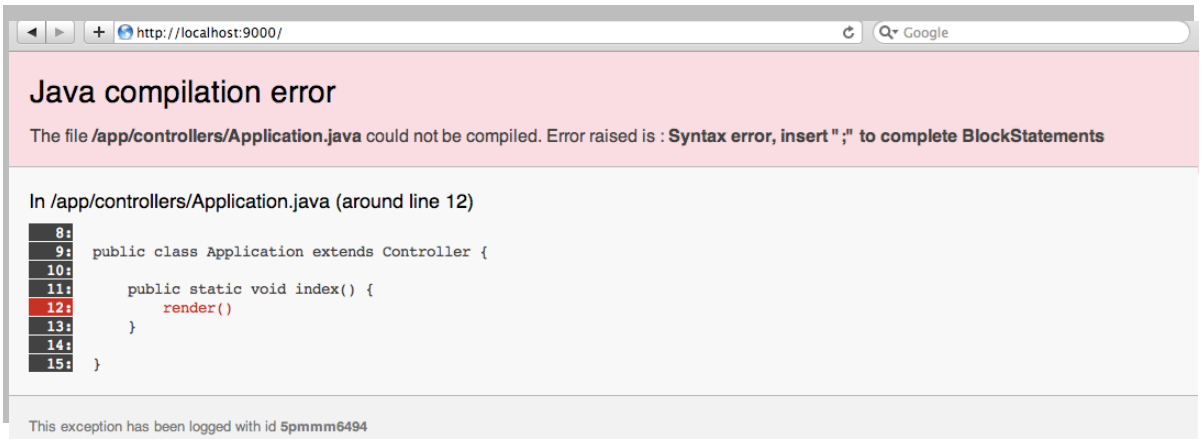

Figura 7: Play Error.

Fonte: Play! . Disponível em: http://www.playframework.com/documentation/2.0/PlayConsole. Acesso em: 15/08/2013.

Seguindo o padrão MVC, o seu projeto consiste de três pastas: *app/**controllers***, *app/**models*** e *app/**views***. Cada controller possui uma pasta em app/views e cada método do controller geralmente possui arquivo template na pasta.

Um dos fatores principais de qualquer framework é persistir dados no banco relacional de forma orientada a objetos. O Play, juntamente com o JPA, resolve esse problema facilmente. Basta criar as classes usando a nomenclatura JPA e realizar as operações para manipulação de dados.

Através do Play, pode-se publicar o aplicativo diretamente para um repositório 'Maven'. A publicação ira enviar o arquivo JAR que contém o aplicativo e o arquivo POM (Project Object Model) correspondente. Outra opção é a publicação na nuvem, algumas plataformas de Cloud Computing já reconhecem, interagem e operam juntamente com o Play. (Ex.: Heroku Cloud Application Platform).

Como uma alternativa limpa e leve para as atuais aplicações Java Enterprise, o Play é focado na produtividade do desenvolvedor e tem por alvo a arquitetura RESTful que garante simplicidade e produtividade.

O Play Framework torna o desenvolvimento de aplicações Java e Scala uma tarefa fácil para o desenvolvedor. Cumpre o que promete, ao conseguir livrar o desenvolvedor de tarefas tediosas como configurar dezenas de arquivos de mapeamento, XML, etc., tornando o desenvolvimento web divertido e produtivo, como a muito tempo não se via em Java.

Existem outros frameworks MVC para desenvolvimento ágil com Java, como o VRaptor e o JSF (JavaServer Faces), porem estes não serão tratados aqui. Existem também, dezenas de outros frameworks que não seguem o padrão MVC, mas que oferecem boa produtividade com Java dependendo da aplicação.

Abaixo, exemplos de aplicações web famosas que utilizam o Play Framework:

Figura 8: Exemplos de aplicações web que utilizam o Play Framework.

Fonte: Play! . Disponível em: http://www.playframework.org/. Acesso em: 10/10/2013.

2.4.2.2. Ruby On Rails (RUBY)

O Ruby é uma linguagem que associa características funcionais e imperativas. O seu criador, Yukihiro “Matz” Matsumoto, uniu partes das suas linguagens favoritas (Perl, Smalltalk, Eiffel, Ada, e Lisp) para formar uma nova linguagem que equilibra a programação funcional com a programação imperativa. Ele disse com frequência que está a “tentar tornar o Ruby natural, não simples”, de uma forma que reflita a vida. Elaborando sobre isto, acrescenta: “O Ruby é simples na aparência, mas muito complexo no interior, tal como o corpo humano”. (Fonte: ruby-lang.org).

Desde que foi tornado público em 1995, o Ruby arrastou consigo programadores devotos em todo o mundo. Em 2006, o Ruby atingiu aceitação massiva, com a formação de grupos de utilizadores em todas as principais cidades mundiais e com as conferências sobre Ruby com lotação esgotada.

O Ruby é visto como uma linguagem flexível, uma vez que permite aos seus utilizadores alterar partes da Linguagem. Partes essenciais do Ruby podem ser removidas ou redefinidas à vontade. Partes existentes podem ser acrescentadas. O Ruby tenta não restringir o programador.

Ruby on Rails é um framework de desenvolvimento web (gratuito e de código aberto) otimizado para a produtividade e diversão do programador. Ele permite que o programador escreva código de forma elegante, favorecendo a convenção ao invés da configuração. (Fonte: rubyonrails.com.br).

David Heinemeier Hansson criou o Ruby on Rails para usar em um de seus projetos na 37signals, o Basecamp. Desde então, passou a divulgar o código e incentivar o uso do mesmo, e em 2006 começou a ganhar muita atenção da comunidade de desenvolvimento web.

O Rails foi criado pensando na praticidade que ele proporcionaria na hora de escrever os aplicativos para web. No Brasil a Caelum vem utilizando o framework desde 2007, e grandes empresas como Abril, Locaweb, Groupon e Twitter adotaram o framework em uma grande quantidade de projetos. (Lista Completa de sites que utilizam o Rails, disponível em: http://www.rubyonrailsgallery.com/. Acesso em: 01/09/2013)

Outro atrativo do framework é que, comparado a outros, ele permite que as funcionalidades de um sistema possam ser implementadas de maneira incremental

por conta de alguns padrões e conceitos adotados. Isso tornou o Rails uma das escolhas óbvias para projetos e empresas que adotam metodologias ágeis de desenvolvimento e gerenciamento de projeto.

Como pilares do Rails estão os conceitos de Don't Repeat Yourself (DRY), e Convention over Configuration (CoC).

O primeiro conceito incentiva o desenvolvedor a fazer bom uso da reutilização de código, que é também uma das principais vantagens da orientação a objetos. Além de poder aproveitar as características de orientação a objetos do Ruby, o próprio framework incentiva o desenvolvedor a adotar padrões de projeto mais adequados para essa finalidade.

O segundo conceito traz o benefício de poder escrever muito menos código para implementar uma determinada funcionalidade na aplicação, desde que o programador respeite alguns padrões de nome e localização de arquivos, nome de classes e métodos, entre outras regras simples e fáceis de serem memorizadas e seguidas. É possível também, contornar as exceções com algumas linhas de código a mais. No geral, as aplicações apresentam um código bem simples e enxuto por conta desse conceito.

A arquitetura principal do Rails é baseada no conceito de separação em três camadas: o MVC (Model, View, Controller). MVC é um padrão arquitetural onde os limites entre seus modelos, suas lógicas e suas visualizações são bem definidos, sendo muito mais simples fazer um reparo, uma mudança ou uma manutenção, já que essas três partes se comunicam de maneira bem desacoplada.

Rails não é baseado num único padrão, mas sim um conjunto de padrões. Outros frameworks que faziam parte do núcleo do Rails antigamente foram removidos para diminuir o acoplamento com o núcleo e permitir que o desenvolvedor os substitua sem dificuldade, mas continuam sendo usados em conjunto. Rails é um "meta-framework" (ou seja, um framework de frameworks), composto por:

- ActionMailer.
- ActionPack.
    - Action View.
    - Action Controller.
- ActiveRecord.
- ActiveSupport.

Por ser uma aplicação em Ruby contendo arquivos .rb e alguns arquivos de configuração diversos, todos baseados em texto puro, o Ruby on Rails não requer nenhuma ferramenta avançada para criar aplicações. Utilizar um editor de textos simples ou uma IDE (Integrated Development Environment) cheia de recursos e atalhos é uma decisão de cada desenvolvedor.

Algumas IDEs podem trazer algumas facilidades como execução automática de testes, syntax highlighting, integração com diversas ferramentas, wizards, servidores, autocomplete, logs, perspectivas do banco de dados etc. Existem diversas IDEs para trabalhar com Rails, sendo as mais famosas:

- RubyMine - baseada no IntelliJ IDEA.
- Aptana Studio 3 - antes conhecida como RadRails, disponível no modelo stand-alone ou como plug-in para Eclipse.
- Ruby in Steel - para Visual Studio.
- NetBeans.

Segundo enquetes em listas de discussões, a maioria dos desenvolvedores Ruby on Rails não utiliza nenhuma IDE, apenas um bom editor de texto e um pouco de treino para deixá-lo confortável no uso do terminal de comando do sistema operacional. Apenas isso já é suficiente para o desenvolvedor ser bem produtivo e não depender de um software grande e complexo para desenvolver. Abaixo, alguns editores de texto conhecidos na comunidade Rails:

- TextMate - o editor preferido da comunidade Rails, somente para Mac;
- SublimeText 2 - poderoso e flexível, é compatível com plugins do TextMate. Versões para Windows, Mac OS e Linux;
- NotePad++ - famoso entre usuários de Windows, traz alguns recursos interessantes e é bem flexível;
- SciTE - editor simples, compatível com Windows, Mac e Linux;
- Vi / Vim - editor de textos poderoso, pode ser bem difícil para iniciantes. Compatível com Windows, Mac e Linux;
- Emacs - o todo poderoso editor do mundo Linux (também com versões para Windows e Mac);
- GEdit - editor de textos padrão do Ubuntu.

Uma das características marcantes do Rails é a facilidade de se comunicar com o banco de dados de modo transparente ao programador.

Dentre todas as opções, uma das mais úteis é a que prepara a aplicação para conexão com o banco de dados. Como o Rails é compatível com uma grande quantidade de bancos de dados, o desenvolvedor pode escolher aquele com o qual tenha mais familiaridade, ou um que já estiver instalado na máquina. Por padrão o Rails usa o SQLite3 pois é bem simples, pequeno, versátil e comporta bem um ambiente de desenvolvimento. As informações podem ser salvas em qualquer local do sistema de arquivos (por padrão dentro da aplicação) e está disponível para Windows, Mac e Linux.

O Rails adota uma estratégia de evolução para o banco de dados (Database Evolution), dessa maneira é possível que a aplicação evolua gradativamente e o esquema do banco de dados seja incrementado com tabelas e campos em tabelas já existentes conforme o necessário. Essa característica vai de encontro às necessidades das metodologias ágeis de desenvolvimento de Software.

Após gerar o código necessário e criar o banco de dados, basta um servidor de aplicações web que suporte aplicações feitas com o Ruby on Rails para colocar a aplicação on-line. Só assim o usuário poderá acessar a aplicação.

O próprio conjunto de ferramentas embutidas em uma instalação padrão de um ambiente Rails já inclui um servidor simples, suficiente para que o desenvolvedor possa acessar sua aplicação através do browser. Esse servidor, o WEBrick, não é indicado para aplicações em produção por ser muito simples e não contar com recursos mais avançados necessários para colocar uma aplicação "no ar" para um grande número de usuários.

Abaixo, exemplos de aplicações web famosas que utilizam o Ruby on Rails:

Figura 9: Exemplos de aplicações web que utilizam o Ruby on Rails.

Fonte: Ruby on Rails. Disponível em: http://www.rubyonrails.com/. Acesso em: 10/10/2013.

2.4.2.3. CakePHP (PHP)

O CakePHP é um framework de desenvolvimento ágil para PHP, livre e de código aberto. Seu principal objetivo é permitir que o desenvolvedor trabalhe de forma estruturada e rápida sem perder a flexibilidade. (book.cakephp.org, 2013).

O framework começou em abril de 2005, quando Michal Tatarynowicz escreveu uma versão mínima de um framework ágil de aplicações em PHP, apelidando-o de Cake. O CakePHP oferece uma estrutura que possibilita aos programadores PHP de todos os níveis desenvolverem aplicações robustas rapidamente, sem perder a versatilidade. O CakePHP utiliza conceitos de engenharia de software e padrões de projeto bem-conhecidos, tais como ActiveRecord, Association Data Mapping, Convenção sobre configuração, Front Controller e MVC (Model-View-Controller).

CakePHP tira a monotonia do desenvolvimento web. O framework fornece todas as ferramentas que o desenvolvedor precisa para começar programando o que realmente deseja: a lógica específica de sua aplicação. Em vez de reinventar a roda a cada vez que se constrói um novo projeto, o programador baixa uma cópia do CakePHP e começa direto com o interior de sua aplicação.

A organização por trás do framework possui uma equipe de desenvolvedores ativa e uma grande comunidade, trazendo grande valor ao projeto. Além de manter o desenvolvedor fora da reinvenção da roda, usar o CakePHP significa que o núcleo da aplicação desenvolvida é bem testado e está em constante aperfeiçoamento.

Principais Características do CakePHP:

- Comunidade ativa e amigável.
- Licença flexível.
- Compatível com o PHP 5.2.6 e superior.
- CRUD integrado para interação com o banco de dados.
- Scaffolding para criar protótipos.
- Geração de código.
- Arquitetura MVC.
- Requisições feitas com clareza, URLs e rotas customizáveis
- Validações embutidas.

- Templates rápidos e flexíveis (Sintaxe PHP, com helpers).
- Helpers para AJAX, JavaScript, formulários HTML e outros.
- Componentes de Email, Cookie, Segurança, Sessão, e Requisições.
- Controle de Acessos flexível.
- Limpeza dos dados.
- Sistema de Cache flexível.
- Localização.
- Funciona a partir de qualquer diretório do website, com pouca ou nenhuma configuração do Apache.

O framework CakePHP utiliza o padrão MVC (Modelo-Visualização-Controlador), porém possui também classes e objetos adicionais que tem como objetivo proporcionar extensibilidade e reuso, para que se possa adicionar funcionalidades à base MVC das aplicações desenvolvidas.

Um estudo detalhado sobre o CakePHP, pode ser acompanhado no site “Cook Book” disponível em: http://book.cakephp.org/2.0/pt/index.html. Acesso em: 09/09/2013. O Cook Book CakePHP é um projeto aberto de documentação editável pela comunidade. Visa manter um alto nível de qualidade, validade e precisão para a documentação do CakePHP.

Outra fonte de consulta é o CakePHP Bakery, que é um centro de intercâmbio para todas as coisas sobre o CakePHP. Locais para tutoriais, estudos de casos e exemplos de códigos. Uma vez que o desenvolvedor estiver familiarizado com o CakePHP, ele pode compartilhar seus conhecimentos com a comunidade e ganhar popularidade e dinheiro com isso. O CakePHP Bakery está disponível em: http://bakery.cakephp.org. Acesso em 09/09/2013.

Abaixo, exemplos de aplicações web famosas que utilizam o CakePHP:

Figura 10: Exemplos de aplicações web que utilizam o CakePHP.

Fonte: CakePHP. Disponível em: http://www.cakephp.org/. Acesso em: 10/10/2013.

2.4.2.4. Catalyst (PERL)

Catalyst é um framework MVC para web, open-source, escrito em Perl, que estimula o desenvolvimento rápido e o design limpo, sem ficar no caminho do desenvolvedor forçando regras. (CATALYSTFRAMEWORK.ORG, 2013).

Catalyst é um framework livre para o desenvolvimento de aplicações web escritas em Perl, que segue à risca o padrão MVC e suporta um grande número de padrões de desenvolvimento web experimentais. Ele é inspirado por outros frameworks web como Ruby on Rails e Maypole. O Catalyst é primariamente distribuído através do CPAN (Compreensive Perl Archive Network), que é o repositório oficial para distribuição de bibliotecas e aplicações escritas em Perl. (MORAIS, 2013).

A web Framework Catalyst é um ambiente abrangente e flexível para criar rapidamente aplicações escaláveis de alta funcionalidade, destinado a projetos web de médio à grande porte. (PERL.ORG, 2013).

Catalyst, em português, significa 'Catalisador', termo muito utilizado na química para substâncias que aceleram reações. A catálise pode ser vista como a mudança de velocidade de uma reação. Os catalisadores agem provocando um novo caminho reacional. Analogamente, assim como na química, o Catalyst existe com o objetivo de acelerar uma reação e prover um novo caminho, neste caso, para o processo de desenvolvimento de software para web.

O framework foi projetado com o desenvolvimento rápido, escalabilidade, extensibilidade, compatibilidade e manutenção em mente. É também baseado no padrão de projeto MVC, para fabricar componentes claramente separados e facilmente intercambiáveis.

“Temos a tendência de manter as coisas pequenas e simples. Isso nos dá robustez e escalabilidade. Seu aplicativo baseado em Catalyst também herda isso. Nada de hierarquias de objeto complicadas.” (CATALYSTFRAMEWORK.ORG, 2013).

Com o núcleo baseado em Moose, o Catalyst é fácil de estender. O framework funciona com todos os principais servidores web. A versão 5.9 tem suporte nativo PSGI / Plack que torna a implantação, para a maioria dos servidores web, intuitiva. Além disso, o Catalyst é capaz de rodar em muitos servidores web diferentes, incluindo mod_perl do Apache e FastCGI.

Catalyst tem um servidor de desenvolvimento integrado. Possui seu próprio servidor de teste leve para o desenvolvimento. O servidor reinicia automaticamente quando algum código fonte é alterado, de modo a obter resultados imediatos.

Quando o desenvolvedor escreve um aplicativo com o Catalyst, ele não precisa se preocupar muito com manipulação de sessão ou autenticação. Caso necessário, o Catalyst faz gerenciamento de sessões, login e autenticação de usuário, caching e internacionalização. Tudo através de plugins testados disponíveis na CPAN (Compreensive Perl Archive Network).

O Catalyst é projetado de modo que cada projeto possa ser rapidamente personalizado para as necessidades específicas desse projeto. Isso dá aos desenvolvedores controle total sobre os requisitos do sistema, utilizando componentes padrões ou substituindo-os caso seja exigida uma solução personalizada. O desenvolvedor pode escolher entre uma grande variedade de modelos (models) e views disponíveis, recorrer ao acervo de plugins, e criar a configuração que se encaixa melhor no projeto.

O framework não dita quais módulos devem ser usados para os modelos (models) e as views, embora a prática padrão atual seja usar DBIx Class para o modelo de banco de dados e Template Toolkit para a view. É fácil adicionar seus próprios módulos, ou usar um dos muitos modelos ou views disponíveis. Catalyst suporta uma ampla gama de models e templates. A criação de versões JSON ou RSS do seu conteúdo requer apenas uma view diferente sobre os seus dados.

Existem outros frameworks para desenvolvimento ágil com Perl, como o Dancer, Jifty e o Mojolicious, porém estes não serão tratados aqui.

Abaixo, exemplos de aplicações web famosas que utilizam o Catalyst:

Figura 11: Exemplos de aplicações web que utilizam o Catalyst.

Fonte: Catalyst. Disponível em: http://www.catalystframework.org/. Acesso: 10/10/2013.

2.4.2.5. Django (PYTHON)

Django é um framework MVC para web, escrito em Python, de alto nível, que estimula o desenvolvimento rápido e limpo, com concepção pragmática. (DJANGOPROJECT.COM, 2013).

O framework tornou-se um projeto de código aberto e foi publicado sob a licença BSD em 2005. O nome Django foi inspirado no músico de jazz Django Reinhardt.

Django foi projetado para lidar com dois desafios: os prazos intensivos de um site jornalístico na cidade de Lawrence, no Kansas e as exigências estritas dos desenvolvedores web experientes que o escreveram. Ele permite criar rapidamente aplicativos web elegantes de alto desempenho. Django framework foca em automatizar o máximo possível da aplicação, aderindo ao princípio DRY (Don't Repeat Yourself). Uma vez que o Django foi desenvolvido em um ambiente dinâmico de uma redação, foi necessário projetá-lo para tornar as tarefas comuns do desenvolvimento web rápidas e fáceis.

Principais Características do Django:

- **Mapeamento Objeto-Relacional**: Com o ORM do Django é possível definir a modelagem de dados através de classes em Python. Com isso é possível gerar suas tabelas no banco de dados e manipulá-las sem necessidade de utilizar SQL (o que também é possível).
- **Interface Administrativa**: No Django é possível gerar automaticamente uma interface para administração para os modelos criados através do ORM.
- **Formulários**: É possível gerar formulários automaticamente através dos modelos de dados.
- **URLs (Localizador-Padrão de Recursos) Elegantes**: No Django não há limitações para criação de URLs elegantes e de maneira simples.
- **Sistema de Templates**: O Django tem uma linguagem de templates poderosa, extensível e amigável. Com ela o desenvolvedor pode separar design, conteúdo e código em Python.
- **Sistema de Cache**: O Django possui um sistema de cache que se integra ao memcached ou em outros frameworks de cache.

- **Internacionalização**: Django tem total suporte para aplicações multi-idioma, deixando o desenvolvedor especificar strings de tradução e fornecendo ganchos para funcionalidades específicas do idioma.

Abaixo, Fig.12, Interface Administrativa do Django.

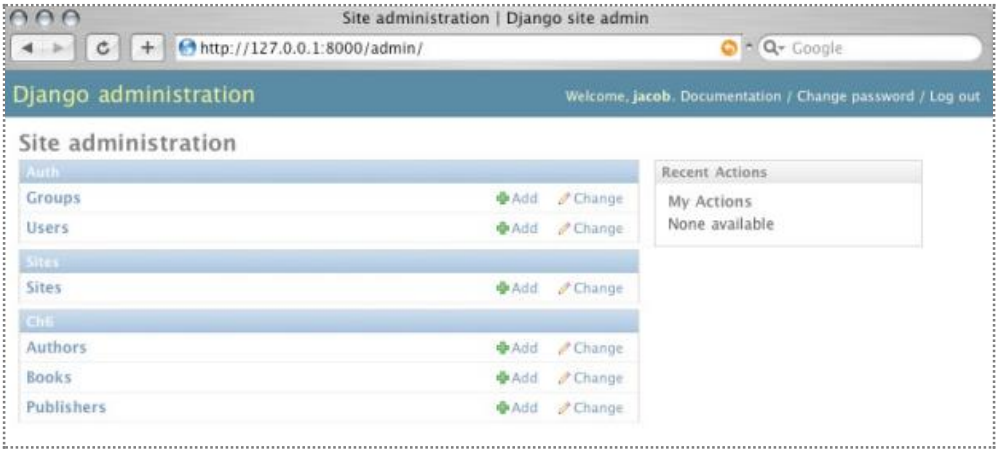

Figura 12: interface Administrativa do Django

Fonte: Django. Disponível em: http://www.djangoproject.com/. Acesso em: 10/10/2013.

Um estudo detalhado sobre o Django Framework, separado em capítulos, pode ser acompanhado no site "The Django Book" disponível em: http://www.djangobook.com/. Acesso em: 09/09/2013. O Django Book é um manual de livre acesso que facilita o trabalho dos programadores exemplificando códigos e fazendo uma análise comparativa entre aplicações escritas em Python sem framework, com as que utilizam o Django. Em geral, as aplicações que utilizam o framework são menos trabalhosas de se programar, levam menos tempo para serem concluídas, são mais simples e fáceis de modificar e manter, além de serem mais escaláveis e seguras. O Django Book foi originalmente publicado pela Apress em 2009 e está sendo atualizado para as novas versões do Django Framework.

Abaixo, exemplos de aplicações web famosas que utilizam o Django:

Figura 13: Exemplos de aplicações web que utilizam o Django.

Fonte: Django. Disponível em: http://www.djangoproject.com/. Acesso em: 10/10/2013.

2.4.2.6. ASP.NET MVC (C#)

O ASP.NET MVC é uma implementação da arquitetura MVC para a plataforma ASP.NET. Constitui um framework com o objetivo de criar aplicações WEB escaláveis no padrão MVC e fornecer uma alternativa (não um substituto) ao modelo webForms do ASP.NET disponível até então. O framework ASP.NET MVC fornece um ambiente robusto e leve que está integrado aos recursos do ASP.NET.
(Fonte: ASP.NET, disponível em: *http://www.asp.net/mvc*. Acesso em: 30/08/2013).

O ASP.NET MVC é mais complexo que o webForms e por isso, exige mais do desenvolvedor. Exige conhecimento e boas práticas de programação. Devido às particularidades do framework, algumas exigências devem ser atendidas ao trabalhar com ASP.NET MVC. Se o uso do framework ainda for uma novidade para a equipe, o tempo disponível para o projeto deverá permitir a aproximação entre framework e equipe.

O ASP.NET MVC exige:

- Equipe madura.
- Equipe disposta ao desafio.
- Equipe motivada a aprender.
- Equipe com cultura de testes.

Durante o fluxo de dados de uma aplicação ASP.NET MVC, o usuário realiza uma requisição a uma página e a aplicação responde com uma ação. Esta ação é um método que estará presente dentro de um determinado Controller. O controller é responsável então, por capturar as informações fornecidas pelo usuário, manipular, acessar o Model e renderizar o conteúdo através de um View.

As requisições feitas pelo usuário são mapeadas para um controlador através das Rotas (Routes). A aplicação verifica qual ação deve ser tomada de acordo com o endereço especificado pelas rotas. Todo esse mapeamento é realizado no arquivo global.asax, que controla todo o ciclo de vida da aplicação.

Abaixo, Fig. 14: Ciclo de Vida de uma Requisição ASP.NET MVC:

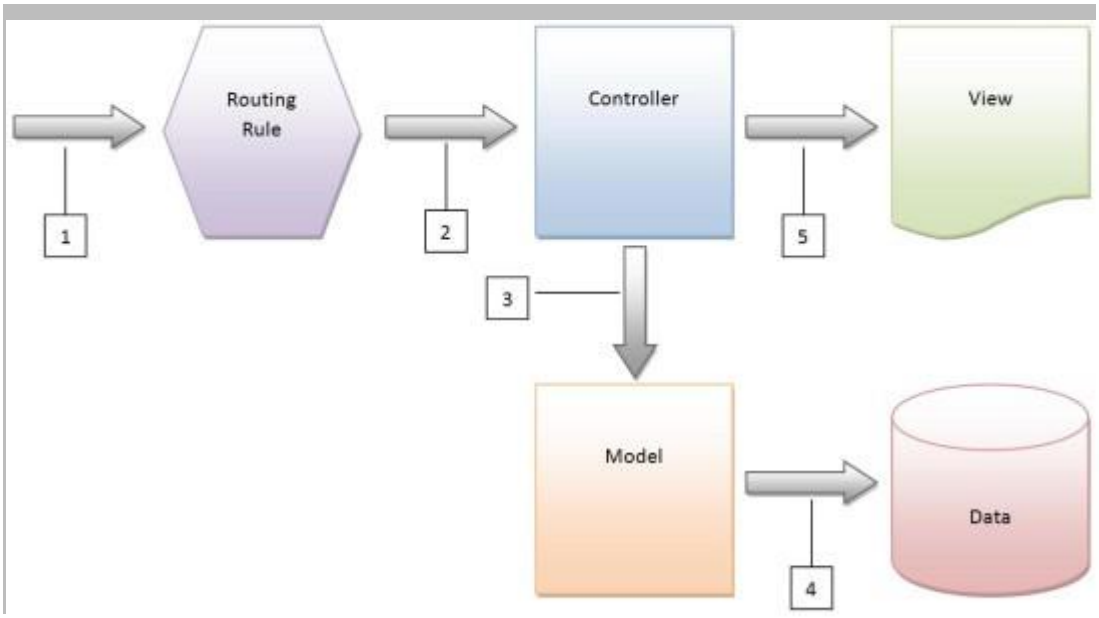

Figura 14: Ciclo de vida de um requisição com ASP.NET MVC.
Fonte: http://dotnetslackers.com/articles/aspnet/KiggBuildingADiggCloneWithASPNETMVC1.aspx. Acesso em: 31/08/2013.

- Usuário Solicita URL.
- ASP.NET MVC Framework avalia a URL solicitada e verifica as regras de roteamento para encontroar o controller correspondente.
- O controlador chama o modelo para construir a View Data. Ele pode chamar o modelo várias vezes, dependendo da View Data que está construindo.
- O modelo, como mencionado anteriormente, é a camada intermediária, que pode envolver componentes de acesso a dados, atividades de fluxo de trabalho, dependências de serviço externo web, etc. O modelo retorna os dados solicitados para Controller.
- O controlador seleciona a View e passa os dados obtidos anteriormente a partir do modelo para o modo de exibição. A View renderiza os dados e retorna o HTML gerado para o usuário.

A seguir, Fig. 15, Template ASP.NET MVC para criação de uma aplicação web utilizando o framework, nas versões anteriores ao ASP.NET MVC 5, no Visual Studio 2008 com .NET Framework 3.5.

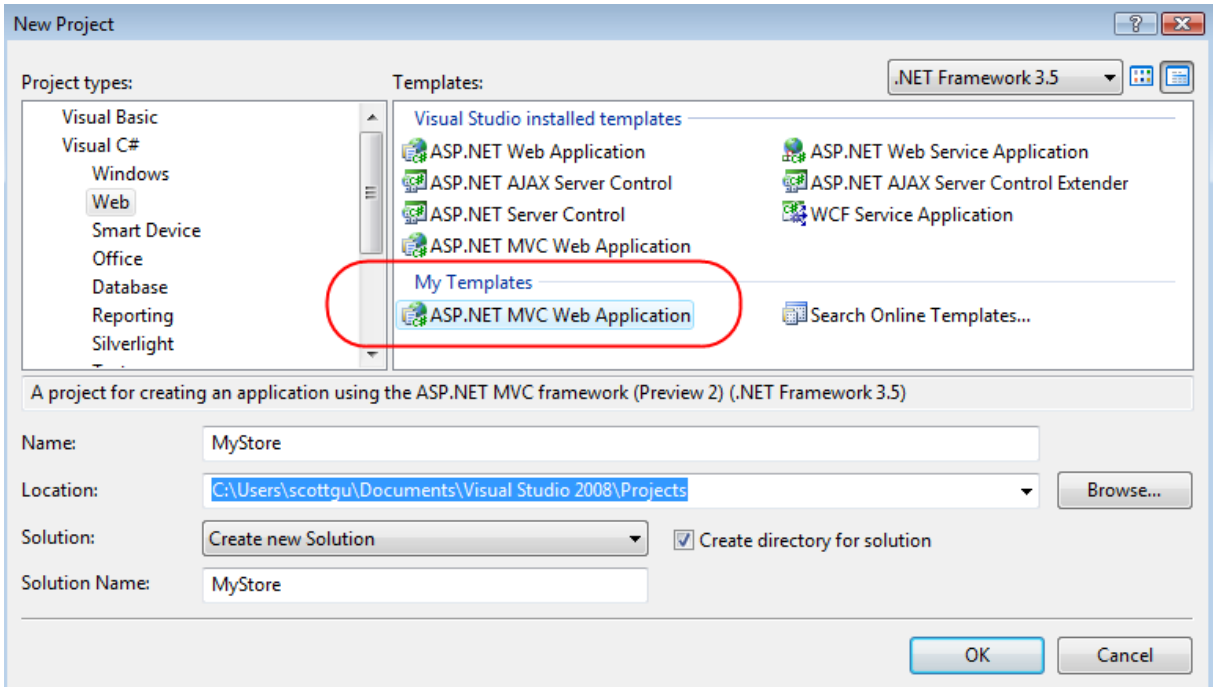

Figura 15: Template ASP.NET MVC.

Fonte: Macoratti. Disponível em: http://www.macoratti.net/13/04/mvc4_app.htm.

Acesso em: 31/08/2013.

Abaixo, Quadros 3 e 4, Vantagens e Desvantagens do ASP.NET MVC Framework. Fonte: Microsoft. Disponível em: http://www.microsoft.com/. Acesso em: 21/10/2013.

Quadro 3: Vantagens do ASP.NET MVC Framework.

Modular	Testável (TDD)
Flexível	Controlável
Manutenível	Evolutivo
Estável	Gerenciável
Padronizado	Open Source
Escalável	DRY (Don't Repeat Yourself)
Extensível	SoC (Separation of Concerns)

Quadro 4: Desvantagens do ASP.NET MVC Framework.

Requer equipe experiente, madura e com conhecimento especializado.
Requer uma quantidade maior de tempo para analisar e modelar o sistema.
Complexo se comparado ao webForms.
Trabalhoso no início do projeto.
Difícil de se integrar com frameworks JavaScript.
Não possui controles com databinding. (Ex.: GridView, DataList...).
Não é aconselhável para pequenas aplicações.

Na programação web, uma das coisas mais entediantes é lidar com erros em um formulário. Mais especificamente, se o desenvolvedor deseja exibir logs de erro, mas precisa manter os dados inseridos anteriormente. O ASP.NET MVC oferece a TempData como um local para armazenar as informações inseridas anteriormente para que o formulário possa ser preenchido novamente. (TAVARES, 2008).

Com o passar do tempo, o ASP.NET MVC Framework deverá fazer parte do controle da interface do usuário, mas é mais provável que sua inicialização não seja tão fácil quanto à dos web Forms, em que inúmeras funcionalidades estavam a uma operação arrastar-e-soltar (drag and drop). Em contrapartida, com o ASP.NET MVC é possível testar a UI (User Interface), o que é praticamente impossível com web Forms que gera código inacessível.

O ASP.NET MVC oferece aos desenvolvedores para a web uma nova forma de criar aplicativos web no Microsoft .NET Framework. A Framework foi projetada tendo em vista a capacidade de teste, adota HTTP, e não tenta abstrai-lo, além de ser extensível a praticamente qualquer ponto.

Trata-se de um complemento atrativo para os web Forms destinado aos desenvolvedores que desejam ter controle completo sobre os aplicativos web.

O ASP.NET MVC 5 foi anunciado no Microsoft Build Developer Conference 2013 (25/06 -28/06), onde foram também anunciadas ótimas novidades para o ASP.Net em geral em conjunto com o novo Visual Studio 2013.

Os templates de projeto ASP.Net MVC 5 integram-se em uma nova experiência de uso chamada One ASP.Net. Agora é possível customizar o template MVC e configurar o tipo de autenticação durante o processo de criação do projeto através do Wizard. Todo projeto ASP.Net MVC 5 agora é uma web Application padrão e não possui um próprio project GUID.

Novidades do ASP.NET MVC 5:

- One ASP.NET
- ASP.NET Identity
- Bootstrap
- Authentication Filters
- Filter Overrides

Mais informações sobre o ASP.NET MVC 5 estão disponíveis em:
http://www.asp.net/vnext/overview/latest/release-notes. Acesso em: 28/08/2013.

Abaixo, Fig.16, Template MVC, para criar aplicações ASP.NET MVC no Visual Studio 2013.

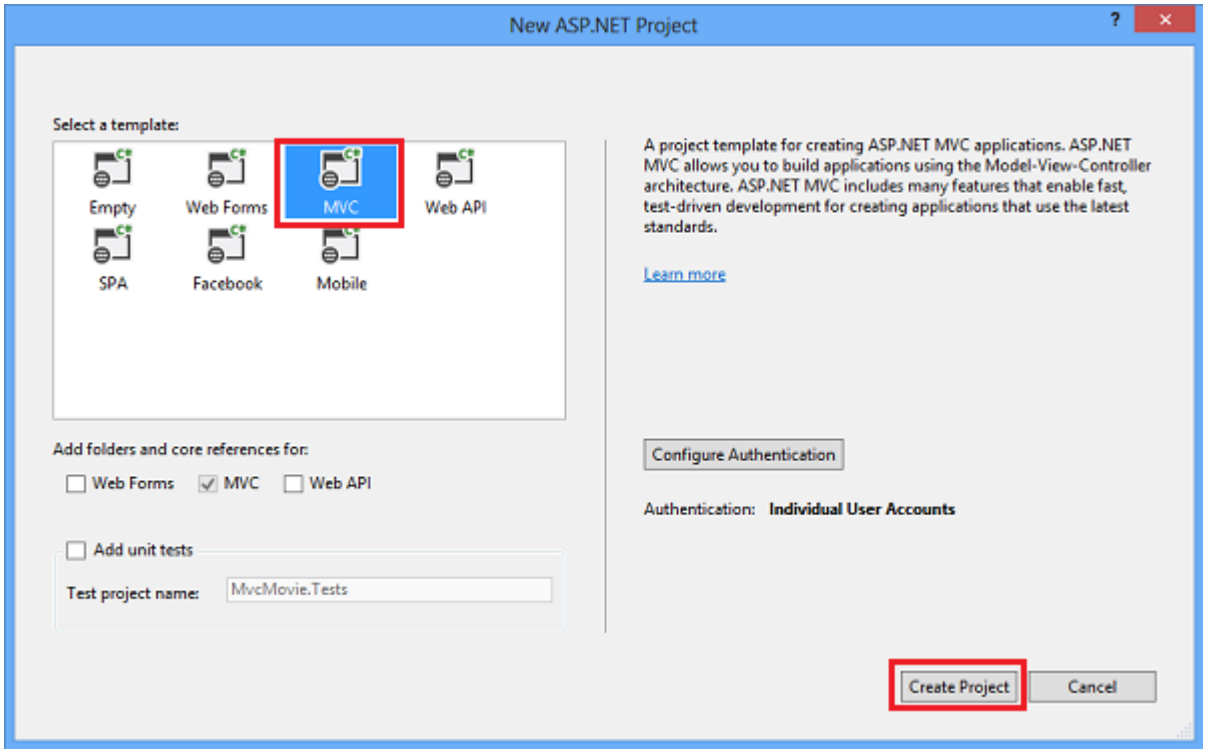

Figura 16: Template MVC.

http://www.asp.net/vnext/overview/latest/release-notes. Acesso em 31/08/2013.

As mudanças em conjunto com o Visual Studio 2013 vão proporcionar mais facilidade e velocidade para criação de aplicações ASP.Net, as melhorias desta nova versão atendem diversas necessidades que eram contornadas de outras maneiras. (PIRES, 2013).

Abaixo, exemplos de aplicações web famosas que utilizam a plataforma ASP.NET:

Figura 17: Exemplos de aplicações web que utilizam ASP.NET.

Fonte: ASP.NET MVC. Disponível em: http://www.asp.net/mvc. Acesso em: 10/10/2013.

2.5. Cloud Computing

2.5.1. Computação em Nuvem

A computação em nuvem refere-se a computadores e aplicativos que são executados remotamente e acessados pela internet. Na computação em nuvem, as máquinas virtuais são executadas em grandes datacenters em substituição aos PCs e servidores físicos. Ao agregar as necessidades computacionais de vários usuários em um único centro, economias de escala são obtidas e resultam em benefícios, como um menor consumo de energia, configuração e manutenção mais simples e atualizações de capacidade e desempenho mais fáceis. Vários aplicativos podem continuar em execução sem passar por alterações à medida que você os reimplanta a partir de desktops locais ou servidores na nuvem. (MICROSOFT, 2000).

A nuvem é um símbolo muitas vezes usado para representar a internet em diagramas e fluxogramas, portanto o termo foi utilizado para representar a internet.

As aplicações da computação em nuvem são praticamente ilimitadas. Com o middleware certo, um sistema de computação em nuvem poderia executar todos os programas que um computador normal rodaria.

Através de tecnologias como o Cloud Computing, a Computação Pervasiva possibilitará a onipresença da tecnologia no cotidiano das pessoas. A interação entre homem e máquina se tornará imperceptível e os dados trafegarão pela nuvem, em grande parte, através do Wi-Fi. Abaixo, Fig.18, esquema de Cloud Computing.

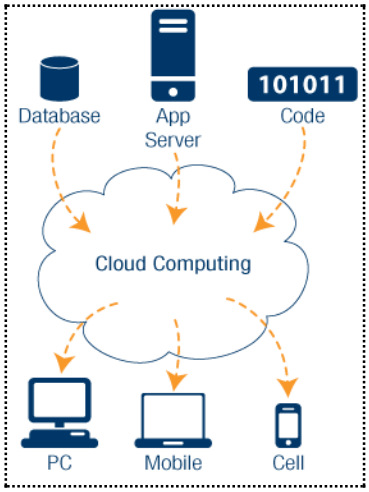

Figura 18: Cloud Computing.

Fonte: Info Escola. Disponível em: http://www.infoescola.com/. Acesso em: 15/08/2013.

Em outras palavras, computação em nuvem é uma tecnologia que permite acesso remoto a programas (softwares), arquivos (documentos, músicas, jogos, fotos, vídeos) e serviços por meio da internet. Basicamente, consiste em compartilhar ferramentas computacionais pela interligação dos sistemas ao invés possuir essas ferramentas localmente. O uso desse modelo é mais viável do que uso de unidades físicas, principalmente no cenário atual onde empresas estão migrando aplicações para a nuvem, usuários estão cada vez mais dependentes do Wi-Fi e a demanda por internet móvel crescendo cada dia mais.

Campeão de reportagens no IDG Now, o Cloud Computing possui diversos modelos de trabalho diferentes, como: IaaS (Infrastructure as a Service), NaaS (Network as a Service), PaaS (Platform as a Service), DaaS (Database as a Service), TaaS (Testing as a Service) e SaaS, (Software as a Service). Este tópico tem foco nas características gerais do Cloud Computing, portanto não aprofundará em detalhes de cada modelo. Apenas os modelos indispensáveis ao desenvolvimento serão discutidos.

O modelo mais referenciado no mercado é sem dúvida o SaaS (Software as a Service). Vender para empresas de todos os tamanhos reduz o custo do software. A empresa fornecedora do software se responsabilizaria por toda a estrutura, licenças e capacitação de pessoas, enquanto a empresa cliente pagaria uma taxa mensal/semestral/anual para utilizar o software via internet. Do software como serviço temos a capacidade de contratar um serviço e pagar somente pelo uso. Essa característica de provisionamento dinâmico é muito interessante, permitindo a redução de custos operacionais, com uma configuração de infraestrutura realmente mais aderente às necessidades de cada negócio.

Abaixo, algumas características do Cloud Computing que motivam clientes (pessoas físicas ou jurídicas) a utilizarem sistemas na nuvem:

- Flexibilidade: Os sistemas na nuvem podem ser acessados de quaisquer dispositivos conectados a internet, independentemente de plataforma. Os dados não estariam confinados no disco rígido do usuário ou na intranet da empresa.
- Menores custos de implementação e manutenção: Sistemas de computação em nuvem reduzem a necessidade de hardwares avançados no lado cliente. Apenas um “terminal burro” ou “thin client” já seria suficiente para acessar os

dados remotos. As atualizações de versão são online e em tempo real. A organização também não precisa se preocupar com licenças de software, ambientes refrigerados para armazenar servidores, e empregados para supervisionar o funcionamento, ao invés disso, a empresa pode optar por pagar uma taxa e terceirizar o serviço na nuvem.

- Escalabilidade: A escalabilidade é uma característica fundamental na computação em nuvem. As aplicações desenvolvidas para uma nuvem precisam ser escaláveis, de forma que os recursos utilizados possam ser ampliados ou reduzidos de acordo com a demanda. O aumento da capacidade de processamento e armazenamento é dinâmico e depende apenas da flexibilidade da aplicação e da renegociação da taxa do serviço. Os riscos com infraestrutura são mínimos. Permite às aplicações fazerem o escalonamento do número de usuários e processamento computacional de modo independente.
- Agilidade: Como quase todo processamento é feito do lado servidor, e as empresas que prestam o serviço na nuvem possuem data centers modernos, servidores robustos e infraestruturas gigantescas, o serviço é entregue ao cliente com mais rapidez.

Desafios encontrados na computação em nuvem:

- Segurança: Um dos maiores desafios a serem enfrentados pela computação em nuvem é a segurança. Nesse novo modelo de computação, o data center armazena informações que os usuários tradicionalmente armazenariam em seus próprios computadores. Além disso, esses usuários desconhecem tanto a localização exata de seus dados quanto à fonte dos dados que estão armazenados junto aos deles (KAUFMAN 2009). Uma aplicação web segura, deve oferecer transparência de acesso (quem) e de localização (onde). No modelo da computação em nuvem, os usuários ficam dependentes da tecnologia oferecida por outros indivíduos ou empresas (WEBER 2008). Portanto, o próprio modelo produz uma grande sensação de insegurança, o que pode acabar afastando os usuários de usá-lo.
- Privacidade: Empresas de computação em nuvem têm a responsabilidade de assegurar a privacidade dos dados de seus clientes. Se um cliente pode logar-se de qualquer local para acessar aplicações, é possível que a

privacidade das informações esteja comprometida. Dessa forma surge a necessidade de utilizar técnicas de autenticação, como usuário e senha, e empregar formatos de autorização (níveis de permissão) onde cada usuário possa acessar apenas os dados e as aplicações que são relevantes para seu trabalho, o que dificultaria a progressão de acesso dentro do sistema por parte dos invasores.

- Disponibilidade: Com o aumento da demanda por serviços de computação em nuvem, as empresas fornecedoras do serviço passam a enfrentar problemas mais complexos de disponibilidade. O ambiente oferecido deve ser resistente a falhas de hardware, software e energia, para manter os serviços disponibilizados o máximo de tempo possível. Com o aumento do tráfego de dados entre servidores e clientes na nuvem, o gargalo da conexão com a internet também pode ocasionar um desempenho ruim na troca de mensagens, dificultando a diferenciação entre falha e atraso na comunicação.
- Governança de TI: Alinhar, ao menos em parte, o conhecimento que a TI e os negócios têm sobre as vantagens e os desafios de se avançar para a nuvem é algo que irá facilitar todo o processo de migração. É necessário que a TI compreenda a linguagem e os desafios do negócio, e o negócio domine ao menos os rudimentos do modelo de computação em nuvem, de modo a não manter expectativas errôneas sobre o que mudará nos processos e nas aplicações da empresa com o avanço para esse novo modelo. É essencial, também, trabalhar para que tanto os membros da equipe de TI como o corpo diretivo da organização conheçam os desafios de se avançar para nuvem. O que norteia a construção de cada um dos níveis e das etapas do planejamento estratégico é a definição prévia dos objetivos a serem conquistados nos processos de negócio e na estratégia de TI. Faz parte também do planejamento estratégico medir de forma precisa, a partir do uso de critérios de avaliação bem específicos, o quanto esses objetivos finais foram realmente atingidos. Naturalmente, quanto mais realista for o objetivo, mais bem sucedido será o projeto e mais cristalinas serão as avaliações do que deu certo e o que teria de ser refeito.

Segundo Mike Small (2011), membro da Associação de Controle e Auditoria de Sistemas da Informação (Isaca), a migração de sistemas para a nuvem necessita

de boa governança de TI. As empresas precisam estabelecer o nível de governança necessário conforme os aplicativos a serem migrados, já que alguns são mais críticos que outros. Abaixo, transição entre modelo tradicional e modelo nas nuvens:

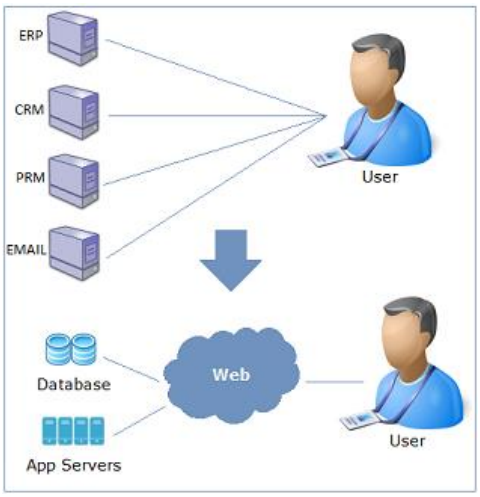

Figura 19: Transição entre Modelos.

No modelo tradicional, o usuário acessa as aplicações através da rede interna da organização. Para isso, o usuário tem que estar fisicamente dentro da empresa, utilizando um computador pessoal. No modelo nas nuvens, o usuário acessa as aplicações remotamente, podendo ou não estar presente na empresa, utilizando os mais diversos dispositivos com acesso à internet.

Computação em nuvem apresenta diversas vantagens, mostra-se um mercado em expansão e possui uma série de desafios a serem superados na utilização desse tipo de ambiente. Sendo alvo de grandes investimentos por grandes corporações, faz com que avanços sejam significativos tornando a computação nas nuvens uma grande aposta para futuro.

A computação em nuvem promete mudar a economia dos datacenters, mas antes que dados confidenciais e regulamentados sejam migrados para a nuvem pública, é necessário tratar de questões relativas aos padrões de segurança e compatibilidade que abrangem a autenticação solida, autorização delegada, gerenciamento de chaves para dados criptografados, proteções contra a perda de dados e emissão de relatórios normativos. Esses são os elementos de um modelo seguro de identidade, informações e infraestrutura e podem ser aplicados a nuvens privadas e públicas e a serviços de IAAS, PAAS e SAAS. (Dorkas, 2009).

2.5.2. Plataformas Afamadas no Mercado

Este tópico tem o objetivo de apresentar plataformas de Cloud Computing conhecidas no mercado e analisar as principais características, pontos positivos e negativos, de cada uma. Três plataformas foram selecionadas com base no grau de aceitação pelo mercado com o objetivo de serem analisadas e comparadas. Existem outras plataformas que não foram citadas aqui que também cumprem o papel de disponibilizar serviços na nuvem.

2.5.2.1. Amazon web Services

Em 2006, a Amazon web Services (AWS) começou a oferecer serviços de infraestrutura de TI para empresas por meio de serviços web, hoje conhecidos como computação em nuvem. Um dos benefícios chave da computação em nuvem é a oportunidade de substituir gastos com a infraestrutura principal por preços variáveis baixos, que se ajustam de acordo com a empresa. (AMAZON, 2013).

A Amazon web Services (AWS) é uma plataforma de computação em nuvem, formada pelo conjunto de serviços computacionais remotos (também chamados de web services), disponibilizados pela Amazon.com. Os serviços mais populares são o Amazon EC2 e o Amazon S3.

Com a AWS o cliente pode calcular o poder de computação, armazenamento e outros serviços relacionados para acessar o pacote de serviços de infraestrutura de TI elásticos como exigido pelo negócio. Com a AWS, o desenvolvedor tem a flexibilidade de escolher qualquer plataforma de desenvolvimento ou modelo de programação que se adapte aos problemas da aplicação em questão. Seguindo o modelo de pagamento por uso, o cliente não possui despesas iniciais ou compromissos de longo prazo, fazendo da AWS a forma mais econômica de distribuição de aplicativos para seus clientes e consumidores.

Além disso, a Amazon.com possui a vantagem da infraestrutura computacional global que é a espinha dorsal da empresa transacional e de vendas multibilionárias, cuja infraestrutura computacional distribuída de maneira segura, confiável e escalável, vêm se aperfeiçoando há mais de uma década.

Atualmente, a Amazon web Services oferece na nuvem uma plataforma de infraestrutura altamente confiável, escalável e de baixo custo que potencializa

centenas de milhares de empresas em 190 países ao redor do mundo. Com datacenters nos EUA, Europa, Brasil, Cingapura, Japão e Austrália, clientes de todos os setores estão tendo vantagens com os seguintes benefícios:

- Baixo custo: A AWS é capaz de construir e gerenciar uma infraestrutura em escala global e passar os benefícios de redução de custos para o cliente sob a forma de preços mais baixos. Com a eficiência de dimensionamento e expertise, a AWS reduziu preços em 15 diferentes ocasiões nos últimos quatro anos. (Fonte: Centro de Informações sobre Economia da AWS, disponível em: http://aws.amazon.com/pt/economics/. Acesso em: 26/08/2013).
- Agilidade e elasticidade instantânea: A AWS fornece uma ampla infraestrutura de nuvem global que permite que o cliente realize mudanças, experimente e interaja rapidamente. Em vez de esperar semanas ou meses pelo hardware, o desenvolvedor pode instantaneamente implantar novos aplicativos, expandir ou reduzir dinamicamente a capacidade computacional à medida que a carga de trabalho aumenta ou diminui. Se o cliente precisar de um servidor virtual ou de milhares, seja por algumas horas ou todos os dias, ainda assim pagará apenas pelo o que utilizar. (Fonte: Centro de Arquitetura da AWS, disponível em: http://aws.amazon.com/pt/architecture/. Acesso em: 26/08/2013).
- Aberto e flexível: A AWS é uma plataforma e um sistema operacional de linguagem independente. O desenvolvedor escolhe a plataforma de desenvolvimento ou o modelo de programação que faz mais sentido para o negócio. O cliente pode escolher os serviços que deseja usar, um ou vários, e escolher como usá-los. Essa flexibilidade permite que o foco do negócio se concentre na inovação, não na infraestrutura. (Fonte: Whitepaper sobre a visão geral da AWS, disponível em: https://d36cz9buwru1tt.cloudfront.net/AWS_Overview.pdf. Acesso em: 26/08/2013.).
- Seguro: A AWS é uma plataforma de tecnologia segura, durável com auditorias e certificações reconhecidas pelo setor: PCI DSS nível 1, ISO 27001, FISMA Moderado, HIPAA e SSAE 16. Os datacenters e centros de serviços têm várias camadas de segurança física e operacional para garantir a integridade e a segurança dos dados. (Fonte: Centro de Segurança da AWS, disponível em: http://aws.amazon.com/pt/security/. Acesso em: 26/08/2013.).

Amazon EC2:

O Amazon Elastic Compute Cloud (Amazon EC2) é um serviço da web que fornece uma capacidade de computação redimensionável na nuvem. Ele foi projetado para facilitar a computação de escala na web para os desenvolvedores. (Amazon, 2013).

A interface simples de serviço da web do Amazon EC2 permite que o desenvolvedor obtenha e configure a capacidade com mínimo esforço. Oferece um controle completo dos recursos computacionais e permite o trabalhe no ambiente computacional comprovado da Amazon. O Amazon EC2 reduz o tempo exigido para obter e inicializar novas instâncias do servidor em minutos, permitindo que o desenvolvedor rapidamente escalone a capacidade para mais e para menos, à medida que os requisitos de computação forem alterados. O Amazon EC2 altera a economia da computação ao permitir que o cliente pague somente pela capacidade que realmente utilizar. O Amazon EC2 fornece aos desenvolvedores as ferramentas para construir aplicativos resistentes a falhas e isolá-los de situações de falha comuns.

Funcionalidade do Amazon EC2:

O Amazon EC2 apresenta um verdadeiro ambiente virtual, permitindo que o desenvolvedor utilize interfaces de serviço web para iniciar instâncias com uma variedade de sistemas operacionais, carregue-os com seu ambiente de aplicativo personalizado, gerencie permissões de acesso da sua rede e execute sua imagem usando o número de sistemas que desejar.

Para utilizar o Amazon EC2, o desenvolvedor deve:

- Selecionar um modelo de Amazon Machine Image (AMI) pré-configurada, para começar a usar o serviço imediatamente ou criar uma AMI contendo as aplicações, bibliotecas, dados e definições de configuração associadas.
- Configurar a segurança e o acesso à rede em sua instância Amazon EC2.
- Escolher os tipos de instâncias desejados, em seguida, iniciar, finalizar e monitorar quantas instâncias de AMI forem necessárias, usando APIs de serviço web ou a grande variedade de ferramentas de gerenciamento fornecidas.

- Determinar se deseja executar em vários locais, utilizar os pontos de extremidade de IP estáticos ou o armazenamento persistente em bloco de conexão para suas instâncias.
- Pagar somente pelos recursos que realmente utilizar como transferência de dados ou instância-hora.

Destaques do serviço:

- Elasticidade: O Amazon EC2 permite que o desenvolvedor aumente ou diminua a capacidade em minutos e não horas ou dias. É possível comissionar uma, centenas ou até milhares de instâncias do servidor simultaneamente. Naturalmente, como tudo é controlado com os serviços de APIs da web, o aplicativo pode automaticamente se expandir ou se reduzir, dependendo da necessidade do cliente.
- Controle: O desenvolvedor tem controle total de suas instâncias. Tem acesso à raiz de cada uma e pode interagir com elas como faria com qualquer máquina. Uma instância pode ser interrompida, mantendo os dados na partição de inicialização, e reiniciada usando APIs de serviços web. O desenvolvedor também tem acesso ao console de saída de suas instâncias.
- Flexibilidade: O desenvolvedor pode escolher tipos de várias instâncias, sistemas operacionais e pacotes de software. O Amazon EC2 permite escolher uma configuração de memória, CPU, armazenamento de instância e tamanho de partição de inicialização que seja ideal para a opção de sistema operacional (várias distribuições de Linux e Windows Server) e aplicativos escolhidos.
- Integração: Projetado para uso com outros Amazon web Services, o Amazon EC2 trabalha em conjunto com o Amazon Simple Storage Service (Amazon S3), o Amazon Relational Database Service (Amazon RDS), o Amazon SimpleDB e o Amazon Simple Queue Service (Amazon SQS) para fornecer uma solução completa de computação, processamento de consulta e armazenamento de uma ampla variedade de aplicativos.
- Confiabilidade: O Amazon EC2 oferece um ambiente altamente confiável, no qual a substituição de instâncias pode ser rápida e previamente encomendada. O serviço é executado dentro da infraestrutura comprovada de

rede da Amazon e Datacenters. O compromisso do Acordo de Nível de Serviço do Amazon EC2 é disponibilidade de 99.95% para cada região do Amazon EC2.

- Segurança: O Amazon EC2 trabalha em conjunto com a Amazon VPC para oferecer funcionalidades de rede seguras e robustas para os recursos de computação alugados.
  - As instâncias de computação estão localizadas em uma Virtual Private Cloud (VPC) com o intervalo de IP especificado pelo cliente. O desenvolvedor decide quais instâncias são expostas para a nuvem e quais permanecem privadas.
  - Os grupos de segurança e os ACLs de rede permitem controlar o acesso de entrada/saída pela rede para/de instâncias.
  - É possível conectar a infraestrutura de TI existente aos recursos da VPC usando conexões VPN IPsec com criptografia padrão do setor.
  - É possível provisionar os recursos de EC2 como instâncias dedicadas. Instâncias dedicadas são instâncias do Amazon EC2 que são executadas em hardware dedicado a um único cliente para proporcionar isolamento adicional.

    Observação: Se o cliente não possuir VPC padrão, deve-se criar uma VPC e executar instâncias nessa VPC para aproveitar os recursos avançados de rede como subnets privados, filtragem de saída por grupo de segurança, ACLs de rede, instâncias dedicadas e conexões VPN.
- Economia: O Amazon EC2 repassa para o cliente os benefícios financeiros da escala da Amazon. O cliente paga uma taxa muito baixa pela capacidade computacional que ele realmente utilizar. O cliente pode economizar através de três tipos de instâncias:
  - Instâncias On-Demand: As instâncias On-Demand permitem que o cliente pague pela capacidade computacional por hora, sem compromissos em longo prazo.
  - Instâncias Reservadas: As Instâncias reservadas dão ao cliente a opção de fazer um pagamento único e acessível para cada instância que ele deseja reservar e receber desconto significativo sobre a taxa por hora para essa instância.

  - Instância Spot: As Instâncias Spot permitem aos clientes negociarem a capacidade não utilizada do Amazon EC2 e executarem essas instâncias durante o período em que sua oferta exceder o Preço Spot atual.
- Usabilidade: O serviço pode ser iniciado a qualquer momento visitando o AWS Marketplace (Fonte: https://aws.amazon.com/marketplace/ref=mkt_ste_ec2. Acesso em: 28/08/2013). Após isso, basta escolher softwares pré-configurados nas Amazon Machine Images (AMIs). A implementação destes softwares no EC2 pode ser feita de forma rápida através da execução com um clique ou com o console do EC2.

O Amazon EC2 fornece um número de características poderosas para construção escalável, falhas de resiliência, aplicativos de classe empresarial, incluindo:

- Amazon Elastic Block Store;
- Instâncias otimizadas para EBS;
- Multiplicidade de locais para instâncias;
- Endereços Elastic IP;
- Amazon Virtual Private Cloud;
- Amazon CloudWatch;
- Auto Scaling;
- Elastic Load Balancing;
- High Performance Computing (HPC) Clusters;
- Instâncias de E/S elevada;
- Instâncias com alta capacidade de armazenamento;
- VM Import/Export;
- AWS Marketplace;

Mais informações sobre os recursos do EC2 estão disponíveis em: http://aws.amazon.com/pt/ec2/. Acesso em: 28/08/2013.

Tipos de Instâncias do Amazon EC2:

- Primeira Geração e Segunda Geração;
- Micro Instâncias;
- Instâncias de mais memória;

- Instância de CPU de alta performance;
- Instâncias de computação em cluster;
- Instância de cluster com mais memória;
- Instâncias de GPU de cluster;
- Instâncias de E/S elevada;
- Instâncias de armazenamento de alta capacidade;

Mais informações sobre tipos de instâncias do EC2 estão disponíveis em: http://aws.amazon.com/pt/ec2/. Acesso em: 28/08/2013.

Abaixo, Quadros 5, 6 e 7. Síntese de Sistemas Operacionais e Softwares aceitos e usados pelo Amazon EC2. Fonte: http://aws.amazon.com/pt/ec2/. Acesso em: 28/08/2013.

Quadro 5: Sistemas Operacionais.

CentOS	Debian	Suse Linux Enterprise
Amazon Linux	Oracle Enterprise Linux	Ubuntu
Red Hat Enterprise Linux	Windows Server	

Quadro 6: Software.

Banco de Dados	Pilhas de Aplicativos
MongoDB	Lamp
Couchbase Server	Tomcat
SAP HANA One	Ruby on Rails
Riak	Django
Microsoft SQL Server	Node.js

Quadro 7: Outros Softwares.

Gerenciamento de Conteúdo	Inteligência de Negócios
WordPress	Jaspersoft Reporting & Analytics
Drupal	MicroStrategy
Joomla!	SAP Business Objects
MediaWiki	Kognito Analytics Platform
Alfresco	

Informações sobre preços de serviços do EC2 estão disponíveis em: http://aws.amazon.com/pt/ec2/. Acesso em: 28/08/2013.

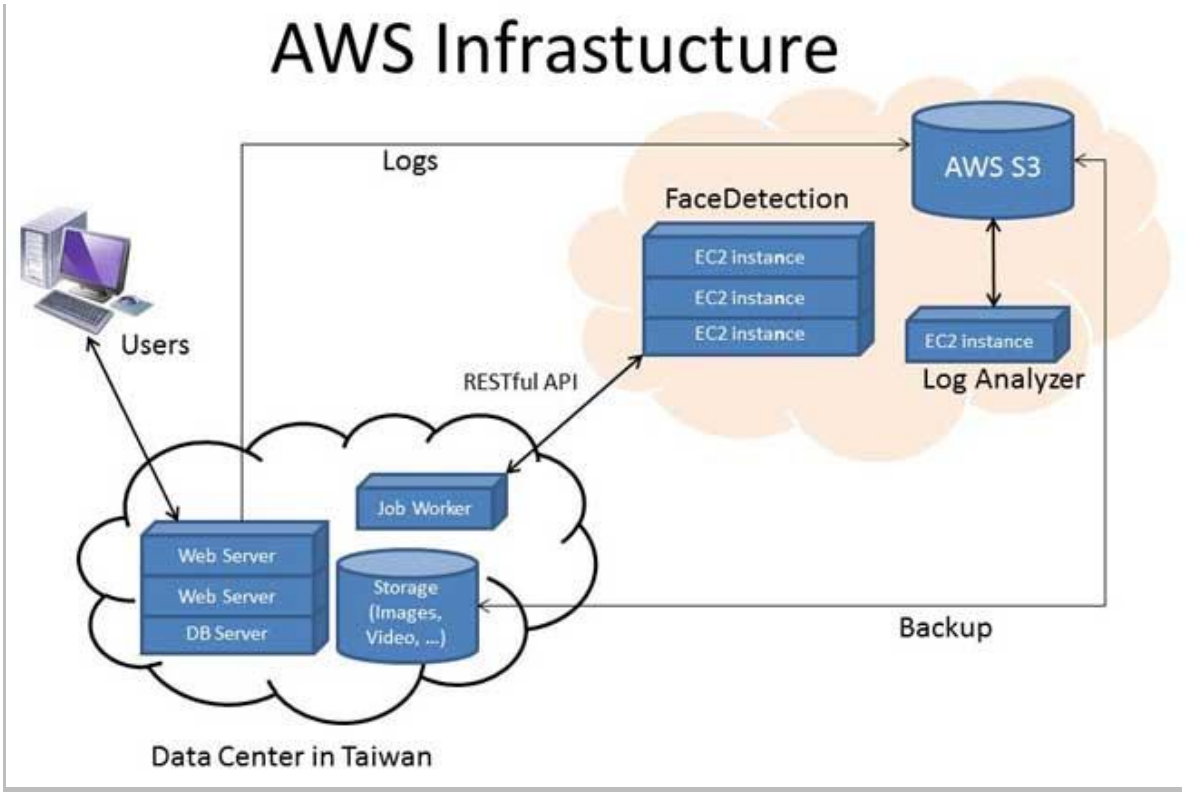

Figura 20: Estudo de caso: PIXNET.

Fonte: Amazon. Disponível em: http://aws.amazon.com/pt/solutions/case-studies/pixnet/. Acesso em: 26/08/2013.

A Fig. 20 apresenta uma solução da Amazon web Services para a PIXNET: Digital Media Corporation (PIXNET). A PIXNET começou a incorporar a AWS em seu sistema usando o Amazon Simple Storage Service (Amazon S3) para repositório de logs diários e backup de arquivos. Pouco depois, a empresa desenvolveu um novo sistema de detecção de rostos e reconhecimento facial para galeria de fotos on-line. Com aumento significativo no número de usuários, a empresa precisava de uma solução de baixo custo que funcionasse em sintonia com os centros de dados existentes sem ter que adicionar nenhum hardware adjacente. Em resposta a este desafio, a PIXNET utilizou o Amazon Elastic Compute Cloud (Amazon EC2) para processar todas as imagens no site da empresa.

2.5.2.2. Heroku

Heroku é uma plataforma de serviço em nuvem (PaaS - Platform as a Service) que roda sobre o Amazon EC2 (IaaS - Infrastructure as a Service) e suporta várias linguagens de programação. Heroku, uma das primeiras plataformas de nuvem, é de propriedade da Salesforce.com e já está em desenvolvimento desde junho de 2007, quando suportava apenas a linguagem de programação Ruby, mas, desde então, adicionou suporte para Java, Node.js, Scala, Clojure, Python e PHP. O sistema operacional de base é Debian ou, no mais recente, o Debian-based Ubuntu.

Se o objetivo é lançar uma aplicação Rails rapidamente, não existe melhor solução. O Heroku automatiza a criação de uma nova máquina virtual e configura todo o ambiente para rodar Ruby.

O Heroku usa uma unidade de máquina virtual chamada “Dyno”. A grosso modo, Dyno é uma máquina virtual pequena, sem swap file e sem suporte a persistência de arquivos. Configurar um novo ambiente é simples, o Heroku possui boa documentação (ver Fig.24) em forma de tutorial que demonstra passo a passo tudo que é possível fazer.

Subir uma única Dyno usando um banco de dados compartilhado PostgreSQL é de graça, o que é excelente para testar aplicações. Nessa modalidade de uso, o banco de dados possui número limitado de linhas (ou tuplas) que podem ser usadas (cerca de dez mil). Obviamente, apenas uma única Dyno não é suficiente para manter uma aplicação em produção para o público.

O Heroku possui uma interface de comunicação direta com frameworks do mercado. Aplicações construídas com Ruby on Rails, Play Framework e Django, por exemplo, podem ser publicadas diretamente no Heroku. A hospedagem possui boa usabilidade para o usuário que pode escalar as aplicações facilmente.

A plataforma Heroku também conta com um acervo de mais de 120 add-ons para acrescentar funcionalidades às aplicações, São aplicativos que oferecem recursos para banco de dados, móbile, busca/pesquisa, log, email, sms, escalabilidade, informações analíticas, caching, monitoramento, multimídia, pagamentos e utilitários. (Fonte: https://addons.heroku.com/. Acesso em: 28/08/2013).

Recursos do Heroku:

- Criação e edição online.
- Importação de aplicações desenvolvidas fora do Heroku.
- Compartilhamento de aplicações.
- Desenvolvimento colaborativo.
- Geração de código (generators e migrate).
- Rails console.
- Ambiente sem necessidade de configuração e deployment.

.

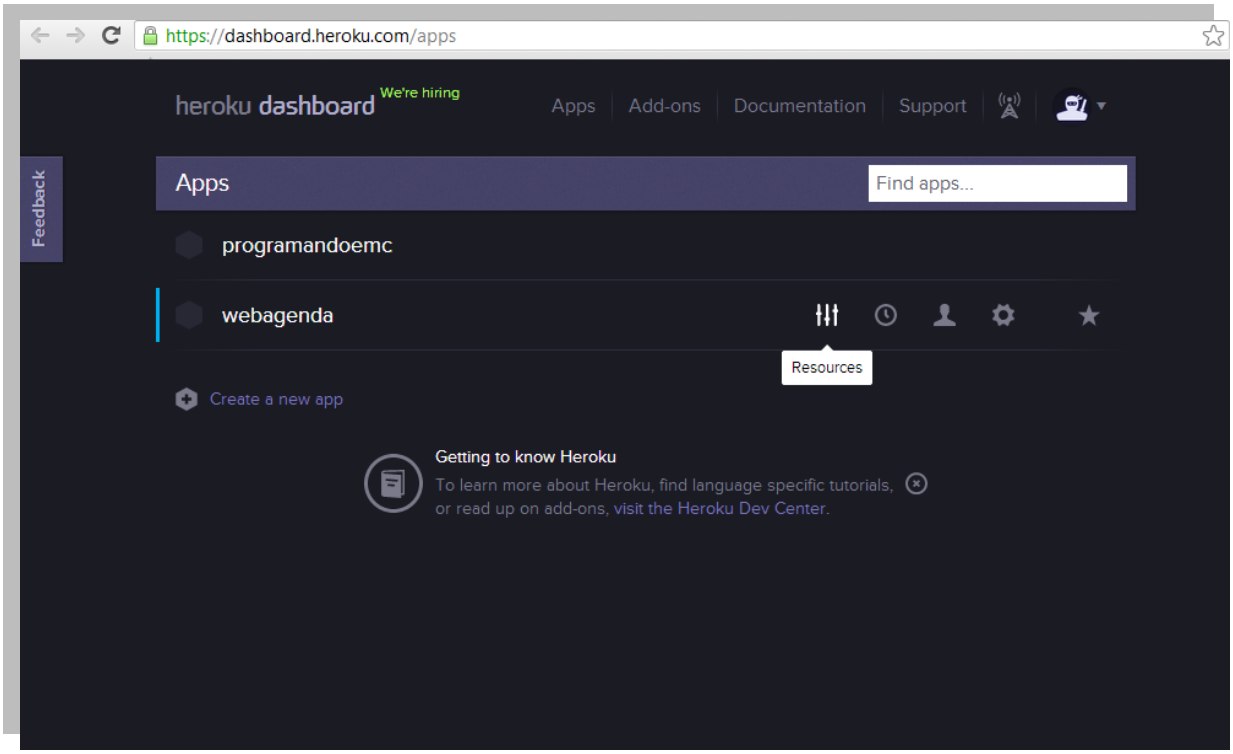

Figura 21: Dashboard Heroku.

Fonte: Heroku. Disponível em: https://dashboard.heroku.com/apps. Acesso em: 04/09/2013.

Após autenticação, o Heroku apresenta uma página (Fig.21) com todas as aplicações hospedadas. O usuário pode inserir novas aplicações, alterar ou deletar aplicações em uso, verificar estatísticas de acesso, quantidade de banco de dados utilizado, valor da taxa mensal, domínio, páginas de erro, etc.

Caso a aplicação web não possua domínio, o Heroku oferece um domínio padrão, gratuito, para endereços. A URL padrão do Heroku, termina com “herokuapp.com”. Exemplo: http://webagenda.herokuapp.com/. (Link ativo, acesso em: 26/08/2013).

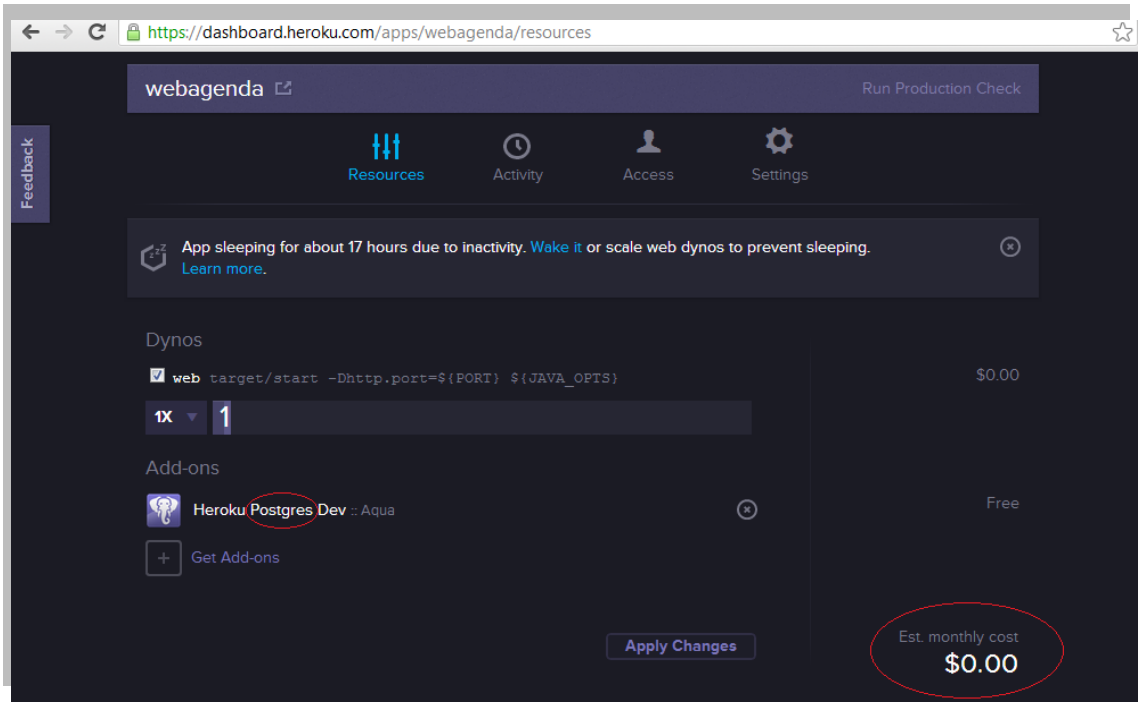

Figura 22: Resourcers da Aplicação.

Fonte: Heroku. Disponível em: https://dashboard.heroku.com/apps/webagenda/resources. Acesso em: 26/08/2013.

A aplicação hospedada no Heroku pode ser escalonada de acordo com a demanda do cliente. A taxa mensal, circulada em vermelho, varia dinamicamente de acordo com a necessidade de expansão ou diminuição do banco de dados (em Gigabytes) e do poder computacional. Na modalidade da Fig.22, o uso é gratuito.

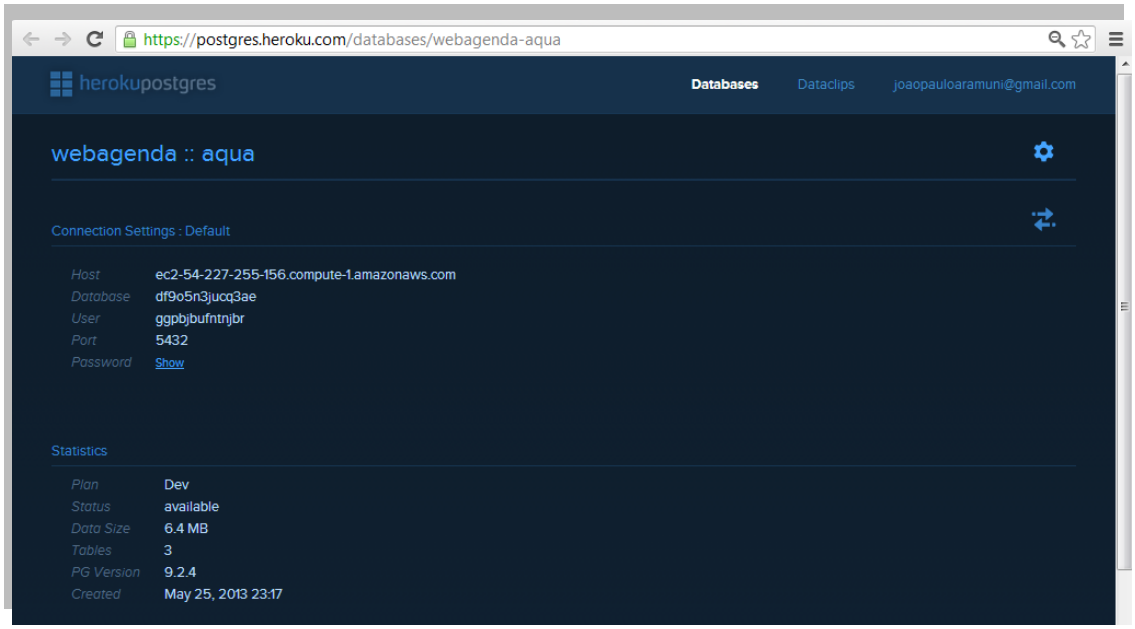

Figura 23: Heroku Postgres.

Fonte: Heroku. Disponível em: https://postgres.heroku.com/databases. Acesso em: 26/08/2013.

A Fig. 23 apresenta a página inicial do Heroku Postgres, com as bases de dados, host, porta, usuário e senha. A página também possui estatísticas com tamanho da base e número de tabelas. Também é possível destruir uma base de dados através das configurações. O Heroku Postgres tem suportado com sucesso 19 bilhões de transações clientes diariamente.

A Heroku mudou a estratégia do seu serviço Heroku Postgres. A partir de agora o serviço provê uma infraestrutura para armazenamento de bases de dados PostgreSQL, independente da solução PaaS (Platform as a Service) utilizada. O Heroku Postgres é uma plataforma disponibilizada com o conceito SQL Database-as-a-Service (DaaS). A proposta é que os desenvolvedores implementem suas aplicações com o PostgreSQL sem se preocupar com questões como disponibilidade e desempenho, uma vez que estas serão resolvidas pela própria infraestrutura do Heroku. (Fonte: Infoq.com. Nov, 2011).

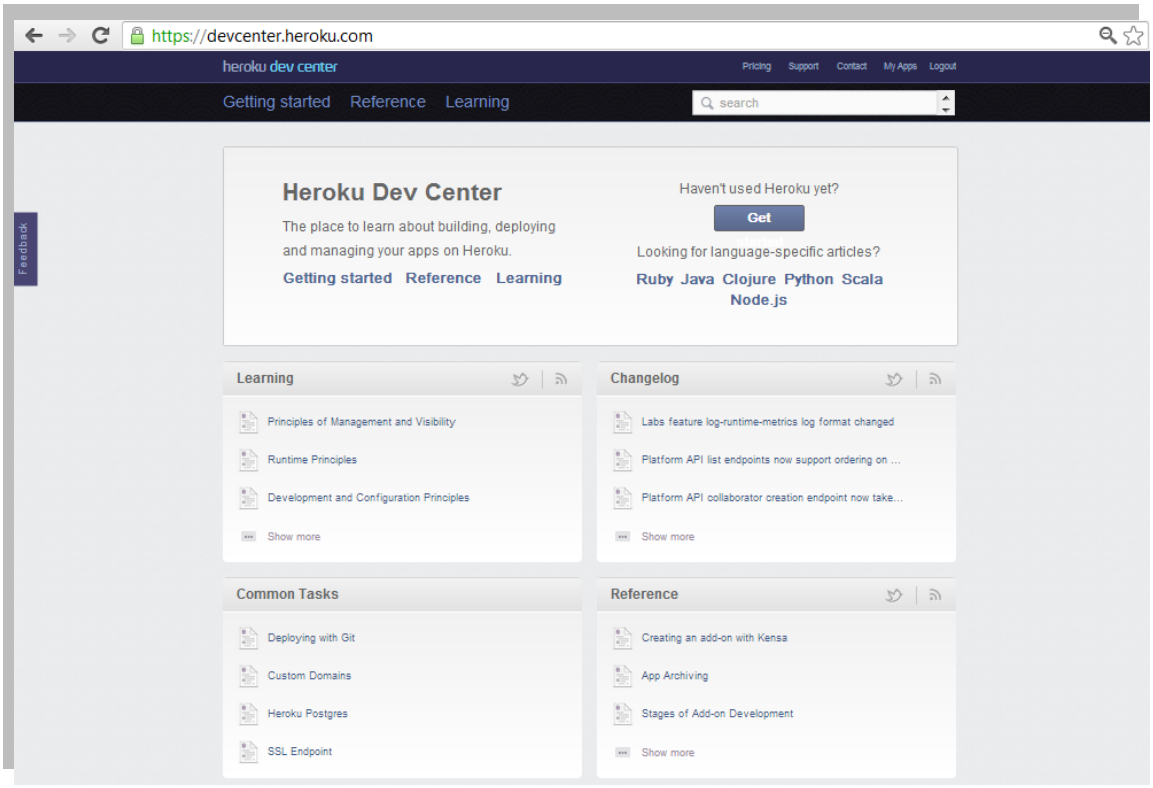

Figura 24: Heroku Dev Center.

Fonte: Heroku. Disponível: https://devcenter.heroku.com/. Acesso: 05/08/2013.

O Heroku Dev Center apresenta documentação extensa e explicativa (Fig. 24) sobre todas as funcionalidades da plataforma. Desde a primeira aplicação publicada, (Getting started) até a manipulação da base de dados (Heroku Postgres).

2.5.2.3. Windows Azure

Apresentado em 27 de outubro de 2008 durante a Conferência de Desenvolvedores Profissionais da Microsoft, o Windows Azure é uma plataforma de alta disponibilidade (acima de 99.9%) para execução de aplicativos na nuvem. É um serviço hospedado e controlado pela Microsoft em seis datacenters distribuídos em três continentes (América do Norte, Europa e Ásia). A plataforma Windows Azure possui as certificações ISO/IEC 27001:2005 e SAS 70 – Certificação Tipo I e Tipo II.

Além dos recursos de computação, armazenamento e administração oferecidos pelo Windows Azure, a plataforma também disponibiliza uma série de serviços para a construção de aplicações distribuídas (PaaS – Platform as a Service), além da total integração com a solução on-premise (local) baseada em plataforma .NET. Entre os principais serviços da plataforma Windows Azure encontramos o SQL Azure Database, (um banco de dados relacional na nuvem), o Azure AppFabric Platform (um barramento de serviços e controle de acesso na nuvem) além de uma API de gerenciamento e monitoração para aplicações colocadas na nuvem.

O público-alvo dos produtos da plataforma Azure são os desenvolvedores. Principalmente desenvolvedores .Net que terão um ambiente integrado para criação, teste e hospedagem de aplicativos na nuvem.

Rompendo os paradigmas de que a nuvem é insegura, o Azure possui diversos modos de segurança, tanto em nível de serviço quanto em nível de aplicativo, incluindo firewalls, roteadores de filtragem, proteção criptográfica de mensagens, gerenciamento de patches de segurança de software, monitoramento e segmentação de rede. O Azure realiza autenticação mútua de SSL para tráfego de controle interno. Todas as comunicações entre componentes internos do Windows Azure são protegidas com SSL. A plataforma foi desenvolvida para ser tolerante a falhas e redundante. (Fonte: Security Overview do Windows Azure).

O Windows Azure oferece opções seguras e flexíveis de desenvolvimento, implantação e dimensionamento para aplicativos web de qualquer tamanho. As ferramentas existentes podem ser aproveitadas para criar e implantar aplicativos sem a dificuldade de gerenciar a infraestrutura. (MICROSOFT, 2013).

Abaixo, os produtos da Plataforma Azure:

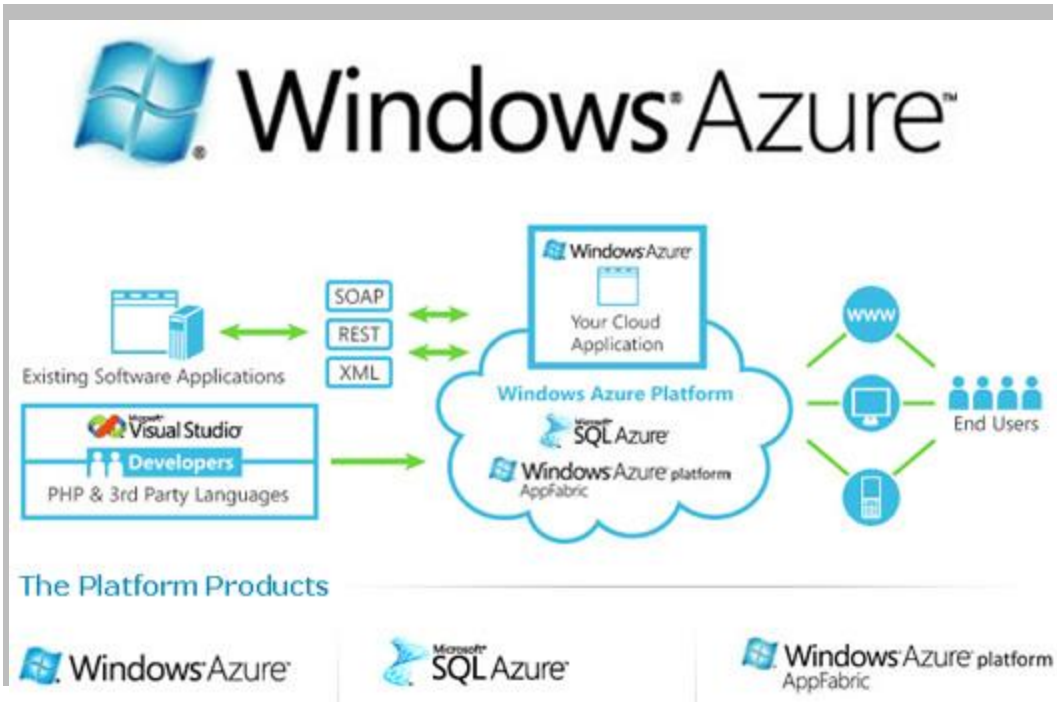

Figura 25: Plataforma Azure.

Fonte: 500Papers.com. Disponível em:

http://goyans.com/home/2013/05/31/windows-azure-steps-by-steps-apps-development/#sthash.ysWPVvZY.dpbs

Acesso em: 25/08/2013.

Com base em publicações oficiais da Microsoft, no Windows Azure webSite, disponíveis em http://www.windowsazure.com/en-us/, a seguir, lista sobre cada produto da plataforma Azure, apresentado separadamente, e os preços por cada tipo de serviço:

Windows Azure:

O Windows Azure fornece recursos essenciais de computação e armazenamento para aplicativos baseados na nuvem. Pode-se usar ferramentas e tecnologias da Microsoft já conhecidas para criar aplicativos, incluindo o .NET, o C++, o ASP.NET, o WCF e o Visual Studio. O Windows Azure fornece a CPU virtual e o armazenamento necessários para tornar os aplicativos mais robustos, permitindo que eles se comuniquem uns com os outros e forneçam os dados armazenados aos usuários finais com eficiência. O Windows Azure pode ser testado gratuitamente.

Síntese sobre Windows Azure:

- Computação: Auto provisionamento de 64 bits; Aplicações residentes em containers em Windows Server VM's; Suporta ampla faixa de modelos de aplicação.
- Armazenamento: Tabelas com alta disponibilidade, blobs e filas com serviços de armazenamento em cache.
- Linguagens: .Net 3.5 (C#, VB.Net, etc.), IronRuby, IronPython, PHP, Java, código nativo Win32.
- Preços (Dados de outubro, 2013):
    - Computação: $0.12 / CPU Hora
    - Armazenagem: $0.15 / GB / Mês - $0.01 / 10k transações / Mês
    - Largura de Faixa: $0.10 in / GB - $0.15 out / GB

SQL Azure:

O SQL Azure fornece recursos de banco de dados relacional, permitindo que os aplicativos armazenem e manipulem dados relacionais em datacenters hospedados pela Microsoft. Também disponíveis estão os Relatórios do SQL Azure (uma versão do SQL Server Reporting Services) e o SQL Azure Data Sync, que permite a sincronização de dados entre bancos de dados locais e baseados na nuvem.

Síntese do SQL Azure:

- Dados: Escalabilidade maciça e bancos de dados relacionais altamente consistentes; Geo-replicação e geo-localização de dados.
- Processamento: Relacional, queries, buscas, relatórios, analítica em dados estruturados, semi-estruturados e não-estruturados.
- Integração: Sincronização e replicação de bases de dados on-premise com outras fontes de dados.
- Preços (Dados de outubro, 2013):
    - Edição web (1GB): $9.99 / Mês
    - Business Edition (10 GB): $99.99 / Mês
    - Largura de Faixa: $0.10 in / GB - $0.15 out / GB

.Net Services:

Serviços .Net oferecem soluções para desenvolvedores que enfrentam problemas para conectar aplicações e serviços com segurança através de localidades diferentes. Os .Net Services oferecem:

- Nuvem-ponte, serviços on-premise e hospedagem de ativos.
- Navegação na rede em limites de segurança, de modo seguro e descomplicado.
- Controle de acesso e identidade através de organizações e provedores de ID.
- Interopere através de linguagens, plataformas e padrões.
- Mediação através de protocolos e mapeamento de esquemas (schema mapping).

Síntese do .Net Services:

- Service Bus: Conectividade de aplicações on-premise; web Services amigáveis, seguros, federados, através de firewalls e mensageria intermediária; Queues de informações duráveis e de fácil descobrimento.
- Controle de Acesso: Identificação federada dirigida por regras; Federação AD; Autorizações baseadas em pedidos.
- Workflows: Orquestração de serviços via atividades baseadas em REST.
- Preços (Dados de outubro, 2013):
    - Barramento de Serviços: $0.15 / 100k mensagens
    - Controle de Acesso: $0.15 / 100k tokens
    - Largura de Faixa: $0.10 in / GB - $0.15 out / GB

Clientes necessitam de uma maneira segura para se conectarem a pontos terminais. Para isso, são necessários barramentos de serviço.

Síntese sobre barramentos de serviços:

- Expõe serviços RESTful ou SOAP na internet através de firewalls e limites NAT.
- Comunicação bi-direcional entre aplicações e serviços por meio de interface inter-operavel.

- Permite escolher relés, filas, roteadores e outros padrões e tipos de mensagem.
- Permite escalabilidade natural e exequibilidade acompanhando crescimento de aplicações e serviços.
- Diminuir barreiras para criação de aplicativos compostos expondo facilmente os terminais, dando suporte a várias opções de conexão e publicar e assinar multicast.
- Ajuda a bloquear tráfego malicioso e protege seus serviços contra invasões e ataques de negação de serviços.

O Service Bus ajuda a fornecer conectividade segura entre serviços mais ou menos acoplados e aplicativos, permitindo-os navegar em firewalls ou nas fronteiras da rede e usar uma variedade de padrões de comunicação. Os serviços que se registram no Service Bus podem ser descobertos e acessados facilmente em qualquer topologia da rede. (MICROSOFT, 2013).

Abaixo, visão geral do Service Bus:

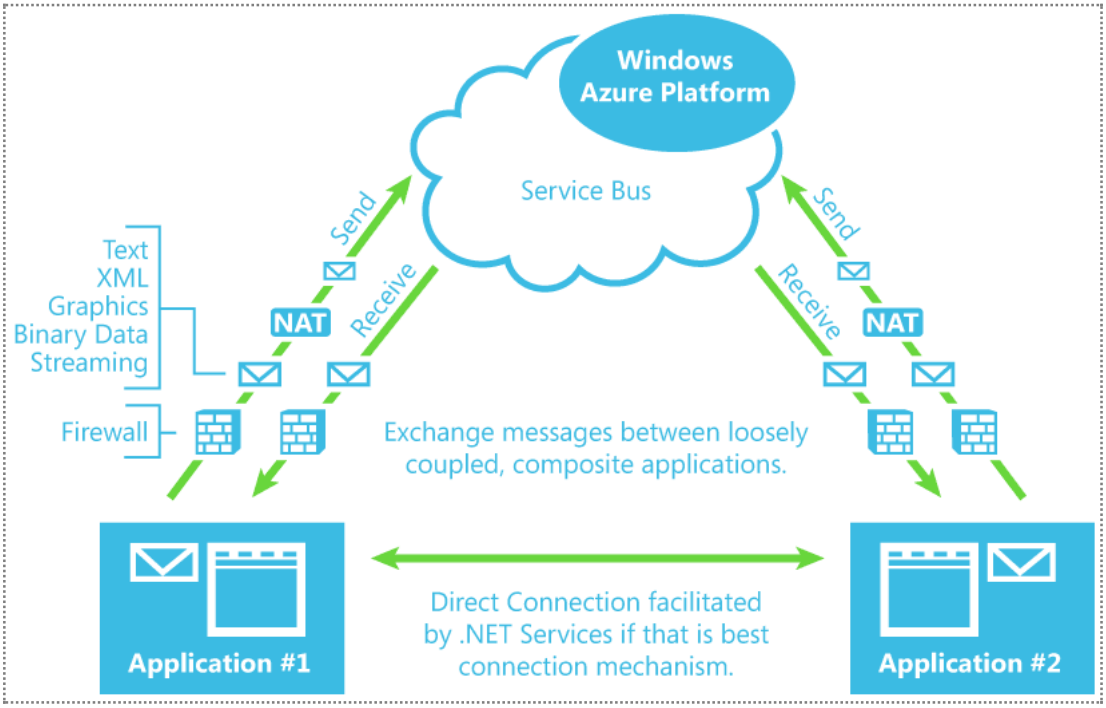

Figura 26: Windows Azure: Service Bus.

Fonte: Heroku. Disponível em: http://www.microsoft.com/brasil/windowsazure/appfabric/. Acesso em: 25/08/2013.

Garantia de Serviço:

- Monitoramento e Reinicialização: Tudo que está rodando é monitorado continuamente. Se o aplicativo ou rotina for corrompido, o Azure detecta imediatamente e toma medidas corretivas (Gerenciamento automático do sistema.).
- Conectividade: O serviço é conectado e acessado via web. Os papeis diversos de acesso na internet tem conectividade externa.
- Disponibilidade de Base de Dados: As bases de dados são conectadas ao respectivo gateway na internet. Monitoramento disponível a cada cinco minutos de intervalo.
- Disponibilidade de Armazenagem: Serviço de armazenamento de fácil acesso, com elevada conectividade. Os pedidos de armazenagem são processados de forma bem sucedida.
- Disponibilidade de Serviços: Os barramentos de serviços (.Net Services Bus endpoint) tem conectividade externa. Solicitações de operação de mensagem serão processadas de forma a serem bem sucedidas.

Características da plataforma Windows Azure:

- Eficiência
  - Solução efetiva em custos para gestão de recursos de TI.
  - Menos infraestrutura para comprar/configurar e dar suporte.
  - TCO (Total Cost of Ownership) mais baixo.
  - Custos previsíveis.
- Inovação / Competitividade
  - Foco na disponibilidade de software eficiente e não em gerenciar infraestrutura.
  - Monetização de novas ofertas rapidamente sem investimento para cobranças ou outras tecnologias habilitadores, como pagamentos.
- Agilidade
  - Velocidade de desenvolvimento.
  - Interoperabilidade.

  - Aproveitamento de IP existente.
  - Disponibilidade simplificada na web.
  - Escalonamento up ou down de acordo com a necessidade do negócio.
  - Acesso rápido ao mercado. (Geração rápida de novos retornos).
- Exequibilidade
  - Serviço exequível.
  - SLAs (Service Level Agreements).
  - Segurança / Redução de Riscos.
  - Centros globais de processamento de dados.

Considerações finais sobre Windows Azure:

O Windows Azure oferece aos seus aplicativos escalabilidade vertical ou horizontal, dependendo das necessidades da empresa. Os desenvolvedores podem usar a criatividade em uma plataforma cuja linguagem eles já conhecem. Seja .Net, PHP, Java ou Ruby, eles terão todo o poder da Nuvem Microsoft. Além disso, com um modelo de pagamento por uso, o cliente não gastará com serviços que pensou que precisaria, mas nunca usou de fato. (MICROSOFT, 2013).

2.6. Segurança

2.6.1. Segurança da Informação

A segurança da informação visa proteger as informações da organização contra os vários tipos de ameaças para garantir a continuidade do negócio, minimizar os riscos, maximizar o retorno sobre os investimentos e as oportunidades de negócio.

A informação é um ativo essencial para a empresa e consequentemente necessita ser adequadamente protegida. Isto é especialmente importante no ambiente de negócios, uma vez que, cada vez mais interconectados, expõem a organização a uma grande variedade de ameaças e vulnerabilidades.

A segurança da informação e o desenvolvimento de software seguro devem ser planejados e projetados de acordo com a expectativa de disponibilidade e confidencialidade dos dados. Um sistema de informação seguro deve ser capaz de impedir que usuários não autorizados tenham acesso aos dados, ao mesmo tempo em que usuários autorizados possam acessá-los.

A informação gerada pelo sistema tem que ser íntegra. Nenhum dado pode ser alterado sem autorização. A integridade e a confidencialidade dos dados devem ser alinhadas e possuir o mesmo grau de importância.

Os sistemas também devem possuir alta disponibilidade, principalmente em ambientes de missão crítica. A disponibilidade do software indica a quantidade de vezes que o sistema cumpriu uma tarefa solicitada sem falhas. Abaixo, Fig. 27: Pilares da segurança da informação.

Figura 27: Pilares da segurança da informação.

Fonte: Claudio Dodt. Disponível em: http://claudiododt.com/2011/06/29/transformando-sua-politica-de-seguranca-da-informacao-em-um-ativo-estrategico/. Acesso: em: 31/08/2013.

Além dos pilares da segurança, outros aspectos devem ser observados como:

- Autenticação: capacidade de garantir que um usuário, sistema ou informação é mesmo quem alega ser.
- Não repúdio: capacidade do sistema de provar que um usuário executou determinada ação no sistema.
- Legalidade: aderência de um sistema à legislação.
- Privacidade: capacidade de manter incógnito um usuário, impossibilitando a ligação direta da identidade do usuário com as ações por este realizadas.
- Auditoria: capacidade do sistema de auditar tudo o que foi realizado pelos usuários, detectando fraudes ou tentativas de ataque. Este aspecto deve ser alinhado à privacidade conciliar os dois é um desafio que requer atenção.

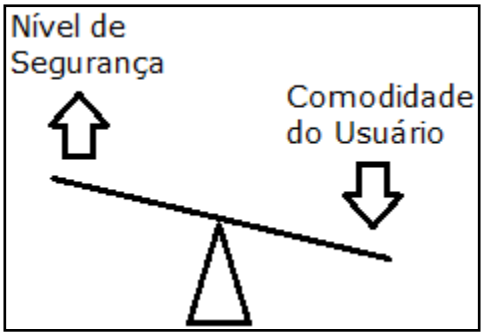

Figura 28: Balança: Segurança de TI.

À esquerda, Fig. 28. Nível de segurança inversamente proporcional à comodidade do usuário. Fonte: Material base da disciplina de Segurança em Redes de Computadores, lecionada pelo Prof. Davis Figueiredo na Universidade Fumec.

Os aspectos apresentados são fundamentais para evitar problemas de segurança e garantir a continuidade do negócio. Esses problemas podem estar relacionados a desastres naturais (tempestades, inundações, furacões, etc...), erros de operação (erro de usuário ou administrador do sistema) e ataques ao sistema. Ataque ao sistema é um tipo de problema de segurança particularmente sério, porque o agente que está realizando o ataque visa obter algum retorno, podendo com isso, lesar a organização e provocar algum prejuízo. Ao contrário dos demais, o ataque pode ser planejado, sistemático e é muito difícil de prever.

O ataque ao sistema tem sempre três elementos fundamentais: o agente que realiza o ataque, um ativo com alto valor e uma vulnerabilidade, ou seja, uma brecha que possibilita atingir o alvo. Usa-se o termo ameaça para indicar a tríade Agente + Vulnerabilidade + Ativo com alto valor, o que resultaria em um ataque.

É importante destacar que as definições para os termos Ataque, Ativo, Vulnerabilidade e Ameaça, foram empregadas de acordo com a ISO/IEC 15.408.

2.6.2. Desenvolvimento Seguro

Existem três preocupações básicas quando falamos de segurança em desenvolvimento de software:

- **Segurança do ambiente de desenvolvimento:** Manter os códigos-fontes seguros, evitar roubo de código-fonte e prevenir a indisponibilidade da equipe de desenvolvimento.
- **Segurança da aplicação desenvolvida:** Desenvolver uma aplicação segura que siga corretamente a especificação de segurança e não contenha acessos ocultos (backdoors), código malicioso ou falhas que comprometam a segurança.
- **Garantia de segurança da aplicação desenvolvida:** Garantir ao cliente a segurança da aplicação em desenvolvimento.

Segurança do ambiente de desenvolvimento: É impossível obter um sistema seguro em um ambiente inseguro. Existem quatro níveis definidos pela ISO/IEC 15.408 para capacidade do ambiente de desenvolvimento e testes de desenvolver uma aplicação segura. Algumas caraterísticas de um ambiente seguro de desenvolvimento:

- Espaço físico restrito, com controle de acesso físico e proteção lógica dos servidores.
- Separação entre ambiente de desenvolvimento, teste e construção (build).
- Gerência de configuração dos códigos-fonte.
- Processos de desenvolvimento bem estabelecidos e controlados, gerando evidencias dos controles.
- Equipe de teste capacitada e equipada para realização dos testes necessários à garantia da segurança.

Segurança da aplicação desenvolvida: Este tópico tem o objetivo de responder algumas perguntas do tipo: Como levantar a necessidade de segurança do cliente? Como transformar isso em uma especificação do sistema? Como implementar essa segurança? E por fim, como testar se o sistema está seguro?

Há um grande numero de recomendações em comum entre segurança da aplicação e normas de boas práticas de programação. De fato, seguindo as normas de boas práticas de programação, muitos erros e falhas de segurança da aplicação serão eliminados automaticamente. São exemplos de normas de boa programação e também práticas para melhorar a segurança de um sistema:

- Funções intrinsecamente seguras: Tratar todas as variáveis de entrada como não confiáveis, verificando sua validade antes usá-las. Isso significa que cada função deve verificar se a variável está entre valores aceitáveis e não contém caracteres estranho ou má formação antes de qualquer uso.
- Sempre verificar os códigos de erro retornados por uma função, principalmente nas chamadas a API do sistema operacional.
- Atentar sempre para o tamanho dos buffers e arrays do sistema. Use strncpy no lugar de strcpy. Ao usar funções de bibliotecas, verificar se não há como estourar um buffer dentro dela.
- Documentar o código: se for trabalho em equipe, é indispensável que não ocorra falha de comunicação entre a equipe.

Para gerar uma aplicação segura, é preciso um ambiente de desenvolvimento seguro, uma boa especificação de segurança e realização dos testes apropriados.

Garantia de segurança não é o mesmo que a segurança do sistema propriamente dita. Fornecer a garantia compreende fazer a aplicação segura e mais alguma coisa que permita ao cliente se assegurar de que o sistema é seguro.

A melhor garantia de segurança para o desenvolvimento de software é através de teste comprovado pelo cliente. Nessa modalidade de garantia, o cliente tem direito a verificar resultados de teste e realizar testes com laboratórios independentes para verificar se a segurança está adequada. Basicamente este é o princípio da ISO/IEC 15.408.

- Especificar a segurança de forma clara e objetiva.
- Construir conforme essa especificação.
- Testar para verificar o atendimento de especificação a segurança.

Tudo isso em um ambiente seguro e seguindo procedimentos bem definidos. A especificação e os resultados dos testes (que podem ser confirmados por laboratórios independentes) são as evidencias que garantem a segurança.

Boas práticas de programação para desenvolvimento de aplicações seguras:

- Criar funções intrinsecamente seguras.
- Usar apenas funções intrinsecamente seguras.
- Testar o retorno de funções chamadas.
- Documentar funções corretamente.
- Verificar caracteres especiais.
- Manter uma política de versão consistente.
- Só usar componentes e bibliotecas confiáveis.
- Evitar informações sensíveis em arquivos temporários.
- Não colocar senhas e chaves de criptografia no código.
- Tratar todas as entradas do sistema como não confiáveis.

A abordagem da ISO 15.408 para a segurança é a da avaliação (ou investigação) da aplicação ou sistema no qual se deseja confiar. Uma avaliação, nesse caso, compreende os meios tradicionais de se assegurar algo, da mesma forma que se avalia um automóvel para assegurar que este é confiável em termos de segurança, ou seja, é necessário avaliar a documentação técnica do veículo e o próprio veículo, de forma a confiar em sua segurança. O Common Criteria propõe, além disso, que a avaliação seja realizada com um incremento progressivo na ênfase dada ao escopo, à profundidade e ao rigor.

Portanto, quando há necessidade de garantir a segurança de uma aplicação, trabalha-se com a premissa de que não existe segurança absoluta para qualquer sistema informatizado, isto é, faz-se necessário fazer com que as ameaças à segurança da aplicação e à política de segurança que esta reforça sejam claramente identificadas e definidas. É preciso também, adotar uma série de medidas com o objetivo de reduzir a probabilidade da ocorrência de vulnerabilidade de segurança. Torna-se fundamental, ainda, analisar medidas que facilitem a identificação e a eliminação de vulnerabilidades que venham a ocorrer.

Uma vulnerabilidade em um sistema pode ser, dependendo do caso, eliminada, minimizada ou monitorada. A presença de vulnerabilidades no sistema pode se dever a erros na definição de seus requisitos, em sua construção ou em sua operação.

Usando a definição mais precisa da ISO, pode-se dizer que garantia é a base para a confiança de que um sistema atinge seus objetivos de segurança definidos. Essa garantia deve ser obtida através de investigação ativa (avaliação) do sistema, de modo a se determinar suas propriedades de segurança.

2.6.3. OWASP

Open Web Application Security Project (OWASP) é um projeto de segurança de aplicativos web open-source. A comunidade OWASP inclui corporações, organizações educacionais e indivíduos de todo o mundo.

Esta comunidade trabalha para criar artigos disponíveis gratuitamente, metodologias, documentação, ferramentas e tecnologias.

A fundação OWASP é também uma organização de caridade, sem fins lucrativos registrada na Europa desde junho de 2011.

A OWASP também é responsável pela criação de padrões emergentes. Teve seu primeiro padrão publicado em dezembro de 2008, a OWASP Application Security Verification Standard (ASVS).

Tem como objetivos criar padrões abertos, adaptados para tecnologias específicas para web. Uma edição para web Services está em desenvolvimento.

Síntese do OWASP (Open Web Application Security Project):

- Promove o desenvolvimento seguro de software.
- Auxilia na tomada de decisão e no gerenciamento de riscos.
- Orientado para o desenvolvimento de serviços baseados na web.
- Focado principalmente em aspectos de back-end.
- Um fórum aberto para discussão.
- Oferece recursos gratuitos e livres para qualquer equipe de desenvolvimento.
- Orientada por esforço voluntário.
- Suportada através de patrocínios.

A OWASP oferece:

- Publicações.
  - OWASP Top 10
  - OWASP Guide to Building Secure web Applications.
- Software:
  - webGoat
  - webScarab
  - oLabs Projects
  - .Net Projects
- Chapters Locais:
  - Orientação das comunidades locais.

OWASP Ferramentas e Tecnologias:

- Automated Security Verification
  - Vulnerability Scanners
  - Static Analysis Tools
  - Fuzzing
- Manual Security Verification
  - Penetration Testing Tools
  - Code Review Tools
- Security Architecture
  - ESAPI
- Secure Coding
  - AppSec Libraries
  - ESAPI Reference Implementation
  - Guards and Filters
- AppSec Management
  - Reporting Tools
- AppSec Education
  - Flawed Apps
  - Learning Environments
  - Live CD
  - SiteGenerator

Organização do OWASP:

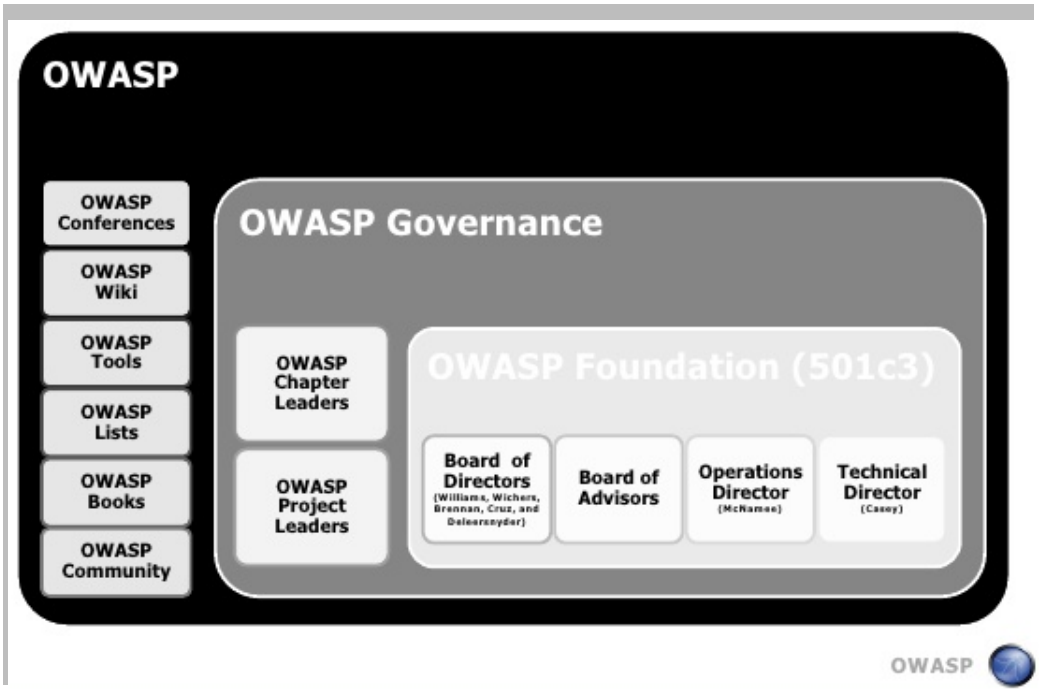

Figura 29: Organização do OWASP.

Fonte: OWASP. Disponível em: http://www.owasp.org/ Acesso: 15/08/2013.

Publicações OWASP:

Todas as publicações OWASP são resultado de trabalho cooperativo entre os membros e estão disponíveis em: owasp.org. As publicações são licenciadas em GNU “Lesser” GNU Public License (LGPL), ou em GNU Free Documentation License (GFDL). A documentação é constantemente atualizada e os projetos são evolutivos.

Top 10 web Application Security Vulnerabilities:

- Uma lista dos 10 aspectos de segurança mais críticos.
- Atualizado em uma base anual.
- Crescente aceitação pela indústria.
  - Federal Trade Commission (US Gov)
  - US Defense Information Systems Agency
  - VISA (Cardholder Information Security Program)
- Está sendo adotado como um padrão de segurança para aplicações web.

Top 10 (versão 2013):

- A1 Injection
- A2 Broken Authentication and Session Management
- A3 Cross-Site Scripting (XSS)
- A4 Insecure Direct Object References
- A5 Security Misconfiguration
- A6 Sensitive Data Exposure
- A7 Missing Function Level Access Control
- A8 Cross-Site Request Forgery (CSRF)
- A9 Using Components with Known Vulnerabilities
- A10 Unvalidated Redirects and Forwards

OWASP Guide. Guia de Desenvolvimento Seguro de web Apps:

- Oferece um conjunto de linhas gerais para o desenvolvimento de software seguro.
    - Introdução à segurança em geral.
    - Introdução à segurança aplicacional.
    - Discussão de áreas-chave de implementação.
        - ✓ Arquitetura.
        - ✓ Autenticação.
        - ✓ Gestão de Sessões.
        - ✓ Controle de Acesso e Autorização.
        - ✓ Registro de Eventos.
        - ✓ Validação de Dados.
- Está em contínuo desenvolvimento.

Projetos em Curso:

- Projeto de Métricas e Medidas: Desenvolve um conjunto de métricas de segurança que podem ser usadas para suportar decisões críticas de negócio.
- Projeto de Testes: Produz um framework de baixo nível para boas práticas que permite identificar certos aspectos.
- AppSec Faq: FAQ para programadores com foco em segurança aplicacional.

Software OWASP: WebGoat

WebGoat é essencialmente uma aplicação de treino. Oferece uma segurança educacional usada para ensinar e aprender sobre segurança aplicacional.

Basicamente é uma ferramenta para testar outras ferramentas de segurança. Esta ferramenta, orientada para o ensino, é uma aplicação web J2EE disposta em diversas “Lições de Segurança”. Utiliza o Tomcat e o JDK 1.5.

O WebGoat tem como características:

- Facilidade de Uso.
- Ilustração de cenários credíveis.
- Demonstração de ataques realistas e soluções viáveis.

O que se pode aprender com WebGoat:

- Um número crescente de ataques e soluções.
  - Cross Site Scripting.
  - SQL Injection Attacks.
  - Thread Safety.
  - Field & Parameter Manpulation.
  - Session Hijacking and Management.
  - Weak Authentication Mechanisms.

O WebGoat está disponível em:
https://www.owasp.org/index.php/Category:OWASP_webGoat_Project

Software OWASP: WebScarab

WebScarab é um framework, open source, extensível de plug-ins, escrito em Java para analisar tráfego HTTP/HTTPS. Permite visibilidade total do protocolo.

Possui múltiplas utilizações, o programador pode realizar o debug das trocas de mensagens entre cliente e servidor enquanto o Analista de Segurança pode estudar o tráfego e identificar vulnerabilidades.

O WebScarab está disponível em:
https://www.owasp.org/index.php/Category:OWASP_webScarab_Project

Chapters Locais do OWASP:

Os Chapters locais proporcionam uma oportunidade para os membros OWASP poderem partilhar ideias e aprender mais sobre segurança da informação. Oferece um local para o desenvolvimento de uma comunidade, onde membros e convidados podem apresentar novas ideias e projetos sobre contextos locais. Esse evento é aberto a todos.

Figura 30: Chapters do OWASP.

Fonte: OWASP. Disponível em: http://www.owasp.org/ Acesso em: 15/08/2013.

O que os Chapters Locais oferecem:

- Reuniões regulares.
- Mailing Lists.
- Apresentação e Grupos.
- Ambientes independentes de Vendedor.
- Fóruns de discussão aberta.

Como contribuir:

- Através das Mailing Lists, reuniões e dos grupos de discussão.
- Levantamento de questões.

- Participação em projetos OWASP.
- Propor novos projetos e lançar iniciativas.
- Manter a organização como um recurso livre, aberto e orientado tecnicamente para o publico em geral e para os membros.

Considerações finais sobre OWASP:

Através das ferramentas oferecidas pelo OWASP, a segurança da informação pode ser maximizada em aplicações web com ganho significativo de tempo e produtividade no desenvolvimento. Com todos os frameworks de detecção e tratamento de ameaças disponíveis, o desenvolvedor pode poupar horas de trabalho e ainda garantir a integridade e confiabilidade do sistema.

Além do ganho de produtividade no desenvolvimento de software, segurança, privacidade e confiabilidade são assuntos que também podem ser alcançados através de uma aliança entre profissionais e frameworks. A balança de trabalho entre framework e desenvolvedor deve ser equilibrada. A segurança da informação não pode ser inteiramente de responsabilidade de um framework, uma vez que essa não é uma ciência determinística e exata.

Sendo assim, o uso de frameworks de segurança da informação deve ser consciente e responsável. Os resultados obtidos devem ser monitorados e acompanhados para garantir a segurança do sistema. Só assim, se firmará uma relação correta entre homem e ferramenta.

A união entre frameworks da OWASP e frameworks de desenvolvimento ágil no padrão MVC, pode contribuir para que o desenvolvedor construa aplicações web rapidamente com qualidade e segurança. O ganho de tempo na fabricação de software é significativo quando todas as tarefas que cercam o processo de desenvolvimento são realizadas de forma otimizada e eficiente.

A OWASP disponibiliza excelentes frameworks, gratuitamente, para dar suporte às equipes que tenham que cumprir curtos prazos de entrega e ainda garantir a segurança da aplicação web desenvolvida.

3. METODOLOGIA

3.1. Definição de Método e Metodologia

O método é um conjunto das atividades sistemáticas e racionais que, com maior segurança e economia, permite alcançar o objetivo, conhecimentos válidos e verdadeiros, traçando o caminho a ser seguido, detectando erros e auxiliando as decisões do cientista. (LAKATOS e MARCONI, 2003, p. 85).

Oliveira (2002, p. 58) contribui, afirmando que método é um conjunto de regras ou critérios que servem de referência no processo de busca da explicação ou da elaboração de previsões, em relação a questões ou problemas específicos.

Método é o conjunto de processos empregados em uma investigação. Segundo Cervo e Bervian (2002, p. 23-25), não inventamos um método, ele depende do objeto da pesquisa, pois toda a investigação nasce de algum problema observado ou sentido, por isso o uso do conjunto de etapas de que se serve o método científico, para fornecer subsídios necessários na busca de um resultado para a hipótese pesquisada.

Segundo Fachin (2003, p. 28), o método científico é um traço característico da ciência aplicada, pelo qual se coloca em evidência o conjunto de etapas operacionais ocorrido na manipulação para alcançar determinado objetivo científico.

O método é, portanto, segundo Oliveira (2002, p. 57), “Uma forma de pensar para se chegar à natureza de um determinado problema, quer seja para estudá-lo, quer seja para explicá-lo.”.

A Metodologia Científica, mais do que uma disciplina, significa introduzir o discente no mundo dos procedimentos sistemáticos e racionais, base da formação tanto do estudioso quanto do profissional, pois ambos atuam, além da prática, no mundo das ideias. Pode-se afirmar que a prática nasce da concepção sobre o que deve ser realizado, e qualquer tomada de decisão fundamenta-se naquilo que se afigura como o mais lógico, racional, eficiente e eficaz. (LAKATOS e MARCONI, 2003, p.17).

O objetivo da Metodologia Científica é tomar os neo-universitários pela mão e caminhar ao seu lado, acompanhando-os em seus primeiros passos de vida universitária, indicando o caminho certo na procura do saber superior, iluminando problemas para que melhor possam vê-los a assumir e a desenvolver hábitos de

estudo e técnicas de trabalho que tornem realmente produtivos os anos de vida universitária, tão preciosos e, não raro, tão mal aproveitados. (RUIZ, 1985, p.16).

A Metodologia estuda os meios ou métodos de investigação do pensamento correto e do pensamento verdadeiro, e procura estabelecer a diferença entre o que é verdadeiro e o que não é, entre o que é real e o que é ficção. (OLIVEIRA, 1997, Apresentação).

3.2. Caracterização do estudo

3.2.1. Segundo os Objetivos

Com base no objetivo geral, esta pesquisa é classificada como **explicativa** e busca registrar fatos, para então sintetizá-los, interpretá-los e identificar suas causas. Essa prática visa ampliar generalizações, definir leis mais amplas, estruturar e definir modelos teóricos, relacionar hipóteses em uma visão mais unitária do universo ou âmbito produtivo em geral e gerar hipóteses ou ideias por força de dedução lógica. Visa identificar os fatores que contribuem para a ocorrência dos fenômenos ou variáveis que afetam o processo em questão.

3.2.2. Segundo as Fontes de Dados

Esta pesquisa científica, quanto à natureza e a fonte de dados, adota modalidade de pesquisa **bibliográfica**. Essa modalidade de coleta pode ser obtida por meio de fontes distintas, tais como literatura científica, publicações periódicas (jornais e revistas), documentos eletrônicos e impressos diversos. Este estudo é embasado em referencial teórico, que fornecerá os conhecimentos empíricos que nortearão o trabalho desenvolvido.

3.2.3. Segundo a Forma de Abordagem

Quanto à forma de abordagem, este estudo caracteriza-se como pesquisa **qualitativa**. Segundo esta perspectiva, um fenômeno pode ser mais bem compreendido no contexto em que ocorre e do qual é parte, devendo ser analisado

numa perspectiva integrada. Vários tipos de dados são coletados e analisados para que se entenda a dinâmica do fenômeno. (GODOY, 1995).

Partindo de questões amplas que vão se aclarando no decorrer da investigação, o estudo qualitativo pode ser conduzido através de diferentes caminhos, fornecendo uma visão panorâmica de três tipos bastante utilizados de pesquisa qualitativa: a pesquisa documental, o estudo de caso e a etnografia.

3.2.4. Segundo o Procedimento de Coleta de Dados

Segundo os procedimentos de coleta de dados, esta pesquisa adota a análise **documental** como técnica para apoiar e enriquecer o referencial bibliográfico.
A análise documental constitui uma técnica importante na pesquisa qualitativa, seja complementando informações obtidas por outras técnicas, seja desvelando aspectos novos de um tema ou problema (LUDKE e ANDRÉ, 1986).

3.2.5. Segundo a Amostra de Coleta de Dados

Os documentos analisados constituem uma rica fonte de dados para estudo.
O exame de materiais de natureza diversa, que ainda não receberam um tratamento analítico, ou que podem ser reexaminados, buscando-se novas e/ou interpretações complementares, constitui a base da pesquisa documental. (GODOY, 1995).

4. CONCLUSÃO

A fabricação de aplicações web de qualidade depende do comprometimento de todos os stakeholders do projeto. O resultado final do software depende não só da equipe de desenvolvimento, mas de todos os envolvidos da empresa responsável pelo software, além do próprio cliente, usuários da empresa contratante do serviço, etc. A governança de TI é essencial neste momento para manter a proximidade entre clientes e fábrica de software. Dessa forma, os erros relacionados a regras de negócio são praticamente anulados.

A relação entre qualidade e tempo é complexa em qualquer tipo de projeto. O aumento da qualidade do software é diretamente proporcional ao tempo para planejá-lo e desenvolve-lo. Uma vez que o tempo se torna um fator precioso para o sucesso do projeto, é necessário pensar em meios eficazes de não desperdiçá-lo. As metodologias ágeis existem justamente para tirar o excesso de burocracia do projeto e melhor aproveitar o tempo disponível. A comunicação entre equipe de desenvolvimento e cliente é de vital importância para ganhar tempo no projeto. De nada adianta utilizar um framework de desenvolvimento ágil se tempo for desperdiçado durante o processo. O cenário ideal para construção de software de forma ágil depende tanto de frameworks de desenvolvimento ágil quanto de metodologias ágeis, ambos com foco no produto, não no processo. Frameworks ágeis possuem enfoque mais prático, enquanto metodologias ágeis possuem enfoque mais teórico. O sucesso de projetos de desenvolvimento ágil de software para web está relacionado a algumas palavras-chave como comunicação, trabalho em equipe, comprometimento e responsabilidade.

Com o crescimento da computação em nuvem, mais aplicações serão demandadas para web e mais complexas serão suas fabricações. A equipe de desenvolvimento será ainda mais exigida e frameworks ágeis terão cada vez mais importância. Em um ambiente onde erros significam prejuízos, mais responsabilidades serão atribuídas aos envolvidos. A segurança por trás destas aplicações, deve obrigatoriamente ganhar espaço durante o planejamento, execução e testes. Qualquer aplicação acessada pela web deve ser exaustivamente testada antes de ser publicada em produção. Uma pesquisa publicada pela White Hat Security mostrou que em 86% dos sites estudados, pelo menos uma falha grave de segurança foi encontrada.

A responsabilidade com a segurança dos dados tem de ser regra em projetos web. A evolução provida pela web dentro da sociedade da informação agrega ainda mais importância aos websites e abre novo mercado para a Engenharia web. Vive-se um momento de novas tecnologias e tendências através da nuvem.

Uma vez que há aumento significativo da demanda por aplicações web, proveniente do cloud computing, faz-se necessário pensar em novas maneiras de se construir software para nuvem. Prazos cada vez menores e procura por mais qualidade e segurança são características presentes em qualquer fábrica de software que desenvolve para web. Dessa forma, frameworks de desenvolvimento ágil apresentam funcionalidades que são de grande importância para o novo cenário da engenharia web.

A escolha do framework ideal para cada tipo de projeto depende de diversos fatores como maturidade da equipe de desenvolvimento, expertise dos profissionais envolvidos, linguagem de programação escolhida pela equipe ou pela empresa, iniciativa dos programadores e cultura/filosofia da empresa sobre o uso de metodologias e frameworks ágeis. Como se trata de um assunto relativamente novo é interessante que as próprias organizações disponibilizem treinamentos sobre o uso de frameworks de desenvolvimento. Esta é uma prática comum em diversas multinacionais pelo mundo.

A equipe de desenvolvimento que deseja começar a utilizar frameworks ágeis precisa, de preferência, se reunir com todos os stakeholders e discutir as vantagens e desvantagens da implantação de um framework no projeto em questão. A equipe deve conhecer suas habilidades e seus limites dentro do seu campo de conhecimento e dentro do próprio projeto.

Os frameworks ágeis apresentados neste trabalho estão entre os mais utilizados por equipes de desenvolvimento e são referência em construção de software para web. A adoção destas ferramentas tem se tornado cada vez mais comum e já é vista no mercado como sinal de modernidade e contemporaneidade. As empresas que entendem esta abordagem e praticam diariamente os preceitos do manifesto ágil acabam destacando-se, gerando competitividade no mercado de TI.

A utilização de frameworks de desenvolvimento ágil agrega valor aos contratos, projetos, e principalmente às pessoas e suas carreiras. A mudança comportamental por trás dos modelos ágeis é o primeiro passo para a evolução gradativa da equipe, bem como do crescimento da própria organização.

5. REFERÊNCIAS BIBLIOGRÁFICAS

5.1. Referências Básicas

AHMED, Mesbah; GARRET, Chris; FAIRCLOTH, Jeremy; PAYNE, Chris. *ASP.NET: Guia do Desenvolvedor web*. 2ed. Rio de Janeiro: Editora Alta Books, 2008, 650 p.

ALBUQUERQUE, R.; RIBEIRO, B. *Segurança no Desenvolvimento de Software*: Como desenvolver sistemas seguros e avaliar a segurança de aplicações desenvolvidas com base na ISO 15.408. Rio de Janeiro: Editora Campus, 2002, 336p.

BECK, Kent; Mike Beedle; Arie van Bennekum; Alistair Cockburn; Ward Cunningham; Martin Fowler; James Grenning; Jim Highsmith; Andrew Hunt; Roland Jeffires; Jon Kern; Brian Marick; Robert C. Martin; Steve Mellor; Ken Schwaber; Jeff Sutherland; Dave Thomas. *Manifesto for Agile Software Development*, 2001. Disponível em: http://www.agilemanifesto.org/. Acesso em: 23/08/2013.

CERVO, Amado Luiz; BERVIAN, Pedro Alcino. Metodologia científica. 5. ed. São Paulo: Prentice Hall, 2002. 242 p.

FACHIN, Odília. Fundamentos de metodologia. 4. ed. São Paulo: Saraiva, 2003. 195 p.

FAYAD, Mohamed; SCMIDT, Douglas; JOHNSON, Ralph. *Building Applications Frameworks*. John Willey, 1999, 688p.

GAMMA, E; HELM, R; JOHNSON, R; VLISSIDES, J. *Design Patterns – Elements of Reusable Object-Oriented Software*. Editora Addison-Wesley, 1995.

GINIGE, Athula e MURUGESAN, San. “Web Engineering: An Introduction”. IEEE Multimedia. Janeiro-Março 2001.

LAKATOS, Eva Maria; MARCONI, Marina de Andrade. *Fundamentos de metodologia científica*. 5. ed. São Paulo: Atlas, 2003. 311 p.

PRESSMAN, Roger S. *Engenharia de Software.* 6ed. Editora Mcgraw-hill interamericana, 2006, 752 p.

SANTOS, Antonio Raimundo. *Metodologia Científica*: a construção do conhecimento. 3ed. Rio de Janeiro: DP&A, 2000.

VELTE, Anthony. *Computação em Nuvem: Uma abordagem Prática*. 1ed. Rio de Janeiro: Editora Alta Books, 2012, 352 p.

5.2. Referências Complementares

BASSI, Dairton. Luiz. *Experiência com desenvolvimento ágil*. Dissertação de mestrado. Disponível em: http://www.teses.usp.br/teses/disponiveis/45/45134/tde-06072008-203515/pt-br.php. Acesso em: 22/08/2013. Instituto de Matemática e Estatística da USP (Universidade de São Paulo). 2008.

DANTAS, Vanessa F. WideWorkweb - Uma Metodologia para o Desenvolvimento de Aplicações web num Cenário Global. Dissertação de mestrado: http://docs.computacao.ufcg.edu.br/posgraduacao/dissertacoes/2003/Dissertacao_VanessaFariasDantas.pdf. Acesso em: 08/09/2013. Centro de Ciências e Tecnologia da UFCG (Universidade Federal de Campina Grande). 2003.

DOUGLAS, Allan. *Por que usar o CakePHP ?*. Artigo disponível em: http://www.devmedia.com.br/por-que-usar-o-cakephp/24437. Acesso em: 04/09/2013.

DRUCKER, Peter. *Managing in the Next Society.* Editora Taylor and Francis, 2007, 248p.

FERREIRA, Alessandro. *Frameworks e Padrões de Projeto*. Disponível em: http://www.devmedia.com.br/frameworks-e-padroes-de-projeto/1111. Acesso: 15/08/2013.

GLASS, R. L. *Who's Right in the web Development Debate?* , Cutter IT Journal, Vol. 14, No. 7, July 2001.

GODOY, Arilda Schmidt. *Introdução à pesquisa qualitativa e suas possibilidades*. RAE - Revista de Administração de Empresas, São Paulo. v. 35, n.2, p. 57-63, 1995. Artigo disponível em: http://www.producao.ufrgs.br/arquivos/disciplinas/392_pesquisa_qualitativa_godoy2.pdf. Acesso em: 29/08/2013.

JOHNSON, Ralph E. Frameworks = (Components + Patterns). Communications of the ACM. Vol. 40. No 10, 1997, p. 39 – 42.

JUNIOR, Evaldo. *Python: O Framework Django*. Artigo disponível em: http://evaldojunior.com.br/blog/2009/10/python-o-framework-django/. Acesso em: 04/09/2013.

KALERMO, Jonna; RISSANEN, Jenni. *Agile software development in theory and practice*. Master's thesis, University of Jyvaskyla, Finland, 2002.

KAUFMAN, L. M. *Data Security in the World of Cloud Computing*. IEEE Security and Privacy, 2009, p. 61 – 64.

LAPWORTH, Leo. *Catalyst – Perl web Framework.* Artigo disponível em: http://www.perl.org/about/whitepapers/perl-webframework.html. Acesso em: 01/09/2013.

LÜDKE, M.; ANDRÉ, M.E.D.A. *Pesquisa em educação*: abordagens qualitativas. São Paulo, EPU, 1986.

MORAIS, Lucas. *Usando o Catalyst Framework Perl.* Artigo disponível em: http://imasters.com.br/artigo/22060/framework/usando-o-catalyst-framework-perl/. Acesso em: 01/09/2013.

MURUGESAN, S; DESHPANDE. Y. Iweb engineering for successful web Application development. In Proceedings of the 21st international conference on Software engineering, 2000.

NIELSEN, Jakob. "Designing web Usability". New Riders Publishing, 2000.

OFFUTT, J. *Quality Attributes of web Software Applications*, IEEE Software, Vol. 19, No. 2, March-April 2002.

OLIVEIRA, Silvio Luiz de. Tratado de metodologia científica: projetos de pesquisa, TGI, TCC, monografias, dissertações e teses. São Paulo: Pioneira Thomson Learning, 2002. 320 p.

PIRES, Eduardo. *ASP.NET MVC 5*: O que há de novo?. Artigo disponível em: http://eduardopires.net.br/2013/07/asp-net-mvc-5-novidades/. Acesso em 31/08/2013.

PREE, Wolfgang; POMBERGER, Gustav; SCHPPERT, Albert; SOMMERLAND, Peter. *Active Guidance of Framework Development. Software - Concepts and Tools*. Springer-Verlager, 1995, p. 94 – 103.

RSA, Dorkas S., Hartman B., Marthers T., Fitzgerald B., Curry S., Nystrom M., Baize E. , Metha N. *O papel da segurança na computação em nuvem confiável*. Acesso em 21/11/2011.

RUIZ, João Álvaro. *Metodologia científica*: guia para eficiência nos estudos. São Paulo, Atlas, 1985.

SAUVÉ, Jacques. *Frameworks e Componentes (PG)*. Disponível em: http://www.dsc.ufcg.edu.br/~jacques/cursos/. Acesso em: 15/08/2013.

SCHMITZ, Daniel. *Redescobrindo Java com Play! Framework*. Artigo disponível em: http://imasters.com.br/artigo/23777/java/redescobrindo-java-com-play-framework. Acesso em: 15/08/2013.

SILVA, Carlos. *Framework CakePHP. Revista PHP.* Artigo disponível em: http://www.revistaphp.com.br/artigo.php?id=93. Acesso em: 04/09/2013.

STEINER, Eduardo. PET Computação: *O Framework Django*. UFSC. Artigo disponível em: http://pet.inf.ufsc.br/sites/default/files/django.pdf. Acesso em: 04/09/2013.

TAVARES, Chris. *ASP.NET MVC*: Criando aplicativos web sem web Forms. Artigo disponível em: http://msdn.microsoft.com/pt-br/magazine/cc337884.aspx. Acesso em 31/08/2013.

THOMAS, Dave. *Desenvolvimento web Ágil com Rails*. 2ed. Porto Alegre: Editora ARTMED, 2008, 680 p.

WEBER, J. Cloud Computing: Are there dangers to having information infrastructure, software and services hosted on the internet rather than on our own personal computers? Revista Times, 5 de maio de 2008.